

University of Alberta

MINIMIZING MOVEMENT IN MOBILE FACILITY LOCATION PROBLEMS

by

Zachary Lorne Friggstad



A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

Department of Computing Science

Edmonton, Alberta
Fall 2007



Library and
Archives Canada

Published Heritage
Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et
Archives Canada

Direction du
Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-33241-2
Our file Notre référence
ISBN: 978-0-494-33241-2

NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.


Canada

Abstract

In the *mobile facility location* problem, which is a variant of the classical uncapacitated facility location and k -median problems, there are clients and facilities starting at locations in a metric space and the goal is to find a destination for each client and facility so that no client is sent to a location that does not receive a facility. The quality of a solution can be measured by either the total distance travelled by all clients and facilities, the maximum distance travelled by any client or facility, or the number of clients or facilities that must move from their original starting location. We examine the complexity of computing or approximating the minimum cost solution to mobile facility location problems. Both algorithms and hardness results are presented to provide a tight classification of the problem under each measure.

Acknowledgements

The list of people who have fostered my academic growth in computing science extends as far back as my high school years. I cannot conceivably give everyone their due thanks in this short note. There are, however, those that distinguished themselves as mentors during my years of study.

Most importantly, I would like to thank my advisor, Mohammad R. Salavatipour. When I began my graduate studies, I had very little knowledge of the rich field of approximation algorithms and he patiently guided me to the understanding I have today. I am indebted to him for his guidance in finding an appropriate problem and for his insight on overcoming some of the obstacles I faced during my graduate studies.

Furthermore, I would like to thank Abdelaziz Fellah, Hua Li, Shelly Wismath, Hadi Kharaghani and Amir Akbary for the research opportunities they presented me during my undergraduate studies. I would also like to thank Howard Cheng and Piotr Rudnicki for teaching me almost everything I know about classical algorithms and their implementations through their roles as coaches for the ACM International Collegiate Programming Competition. Thanks also to Phillip Henderson for patiently listening to some of my ideas when they were in their initial stages of development.

It is impossible for me to express my gratitude to my wife for the overwhelming love and support she has given me over the last few years. I also thank my parents for their wisdom and love they have been graciously giving me my whole life. Knowing that it is futile to try and express the whole of my appreciation for this, I thank God for the honour and blessing of being a husband, son and brother in my family.

Finally, I would like to thank NSERC, iCORE and the University of Alberta as a whole as well as its Department of Computing Science for their financial support during my studies.

Table of Contents

1	Introduction	1
1.1	The Problem and its Motivation	1
1.2	Notations and Preliminaries	3
1.3	Previous Work	7
1.4	New Results	10
2	Initial Results	12
2.1	Solving CM-MFL	12
2.2	A 2-Approximation for MM-MFL	14
2.3	A Pseudo-Polynomial-Time Exact Algorithm for TM-MFL in Trees	14
3	Hardness Results	20
3.1	MM-MFL is $(2 - \epsilon)$ -Hard to Approximate	20
3.2	TM-MFL is APX-Hard	21
3.3	Solving the k -Median Problem with TM-MFL	22
4	An 8-Approximation Algorithm for TM-MFL	24
4.1	Overview	24
4.2	A Randomized 16-Approximation Algorithm	27
4.2.1	Clustering of Clients	27
4.2.2	Relocation of Facilities	29
4.2.3	Getting a Half-Integer Solution	31
4.2.4	Modifying the Half-Integer Solution	33
4.2.5	Randomized Rounding	36
4.2.6	Putting it all Together	38
4.3	Improving to a Deterministic 8-Approximation	38
4.3.1	Improving Analysis in the Relocation of Coverage	38
4.3.2	Improving the Expected Cost	39
4.3.3	Derandomizing the Algorithm	40
4.3.4	The Final Analysis	41
4.3.5	The Running Time	41
4.4	A Bicriteria Approximation Algorithm	43
5	Conclusion	46
5.1	Summary	46
5.2	Directions for Future Work	47
	Bibliography	49

List of Tables

1.1	Known approximation ratios & lower-bounds for movement problems. All results, unless marked as new, are found in [13].	7
-----	--	---

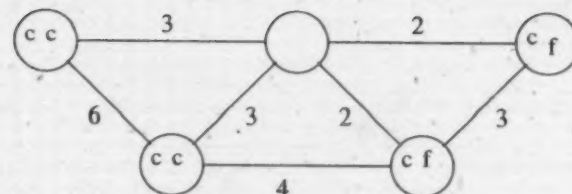
Chapter 1

Introduction

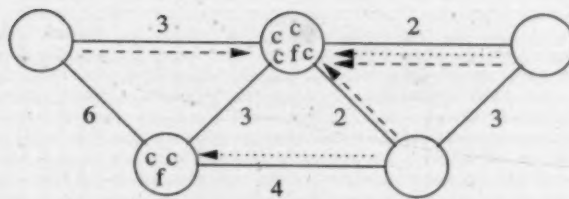
1.1 The Problem and its Motivation

The general setting of bringing clients and facilities together in a cost-efficient way has received much attention lately. These problems typically involve clients and facilities being located in a metric space where the goal is to make sure that each client is being supplied by some facility while minimizing a specified cost function. In many such problems, computing a feasible solution that minimizes the cost function is proven to be NP-hard. Thus, if one wants to find a feasible solution to any given instance of the problem in a reasonable amount of time, it should be expected that this answer is sub-optimal. An approximation algorithm will quickly (e.g. in polynomial time) compute a feasible solution to an optimization problem whose cost is within a certain factor of the minimum possible cost.

The client/facility model that we consider in this thesis is the following. We are given a metric space consisting of a finite set of locations V and distances d_{uv} between any $u, v \in V$. The distances are symmetric, non-negative and satisfy the triangle inequality $d_{uv} \leq d_{uw} + d_{wv}$ for all $u, v, w \in V$. For simplicity of presentation, we allow $d_{uv} = 0$ for distinct locations $u, v \in V$. Scattered throughout the locations are two types of pebbles which we distinguish as *clients* C and *facilities* F . The goal is to move some or all of the pebbles to new locations so that no client pebble is on a location that is not the location of a facility pebble. The three ways to measure the quality of such a solution that we consider are the total distance moved by all pebbles, the maximum distance that any pebble has to move, and the total number of pebbles that were moved from their original start locations. For example, minimizing the total amount moved by all pebbles can correspond to minimizing the total amount of resources (e.g. gasoline) consumed in moving all clients and facilities to their destinations. Minimizing the maximum that any pebble moves can be thought of as minimizing the amount of time it takes to simultaneously move all pebbles to their final locations (e.g. response time). Finally, situations where dislodging clients or facilities from their current locations is the dominant cost (e.g. the setup cost dominates the cost of travel) can be modeled by considering the total number of pebbles that must move as the cost of the solution. See Figure 1.1



(a)



(b)

-----> client pebble movement
> facility pebble movement

Figure 1.1: An instance of the problem on a graph (a) and a solution (b) that moves 6 pebbles with maximum movement 4 and total movement 16. The optimal solutions for the number of pebbles moved, the maximum any pebble moves and the total distance travelled by all pebbles are, respectively, 4, 3, and 16.

for an example of the problem and a possible solution.

The client-facility model has many immediate applications. Consider the following scenario. There is a company with some manufacturing plants. There are also several retail stores to which these products must be shipped. One possibility is to send the products to each retailer from its closest manufacturing plant. However, it may be cheaper to set up a few distribution centers. Each facility will send their product to a distribution center and each retailer will pick up these products from their nearest distribution center. This model corresponds to the problem we consider in this thesis. Many such models have been considered heavily in operations research and include the famous *uncapacitated* and *capacitated facility location* problems as well as the *k-median* problem. As we shall soon see, the total-movement variant can be thought of as one of the natural ways to generalize both the uncapacitated facility location problem and the *k-median* problem.

In general, movement problems have a wide variety of applications. Demaine *et. al.* [13] introduced this class of problems in which we are typically given an instance involving a weighted graph G together with pebbles on the vertices (and/or edges) and a desired property P . The goal is to move the pebbles around the graph so that in the end, the property P is satisfied. The natural cost functions they consider are the total distanced travelled by all pebbles, the maximum distance any pebble must move, and the number of pebbles that must move from their start location. Many

problems of this type arise naturally in other fields, such as operations research, robotics, and the design of systems of wireless networks. For instance, suppose each pebble corresponds to a wireless sensor and our goal is to move these sensors around so that they form a connected network. This corresponds to the movement problem with property P being the subgraph induced by the pebbles' final locations being connected. See, e.g., [6, 21] and the references in [13] for examples of research in this scenario and other applications that can be modeled as a movement problem.

1.2 Notations and Preliminaries

The client/facility model we consider in this thesis is called the *mobile facility location* problem or MFL. The problem of minimizing the total distance moved by all pebbles is the *total-movement mobile facility location* problem and will be denoted TM-MFL. The problem of minimizing the maximum distance moved by a pebble is called the *maximum-movement mobile facility location* problem and will be analogously denoted MM-MFL. The problem of moving the least number of pebbles is the *cardinality-movement mobile facility location* problem and is denoted CM-MFL.

A graph G is a pair (V, E) where V is a finite set of vertices (nodes) and E is a collection of edges joining pairs of vertices. If $e \in E$ joins vertices u and v , we say that u and v are the endpoints of e . A *directed graph* is one where each edge $e = (u, v)$ has an orientation where one of its endpoints is called the *tail* of the edge and the other is the *head* of the edge. A *weighted graph* is a (possibly directed) graph $G = (V, E)$ with a weight function $w : E \rightarrow \mathbb{R}$. Given a subset U of V , the subgraph of G induced by U is the graph $G' = (U, E')$ where $E' = \{(u, v) \in E : u, v \in U\}$. A *walk* in a graph is a sequence of vertices $v_1, v_2, \dots, v_k \in V$ such that $(v_i, v_{i+1}) \in E$ for all $1 \leq i < k$. In directed graphs, we must have that each (v_i, v_{i+1}) edge used in a walk has v_i being the tail of that edge and v_{i+1} being the head of that edge. If the graph is weighted, we say that the *length* of this walk is $w(v_1, v_2) + w(v_2, v_3) + \dots + w(v_{k-1}, v_k)$. A *path* is a walk where each vertex is visited at most once and a *shortest path* between vertices u and v in a graph is a path starting at u and ending at v of minimum length where we say all edge weights are of unit length in an unweighted graph. A *cycle* is a walk $v_1, v_2, \dots, v_k, v_{k+1}$ where $v_1, v_2, \dots, v_k$ is a path and $v_{k+1} = v_1$. A *connected component* of a graph is a maximal (with respect to inclusion) set $U \subseteq V$ such that there is a path between any two vertices $u, v \in U$.

A *bipartite graph* is a graph $G = (V, E)$ where the vertices of V can be partitioned into two disjoint sets V_1 and V_2 (called *bipartitions* of G) such that the two endpoints of any edge $e \in E$ lie in different sets. A *matching* in a graph is a set of edges $M \subseteq E$ such that each vertex is the endpoint of at most one edge of M . A *perfect matching* is one where all vertices are the endpoint of some edge of M , or $2|M| = |V|$. Note that a perfect matching in a bipartite graph is only possible when $|V_1| = |V_2|$. The weight of a matching M is the sum of all the weights of edges in M . Finally, a complete bipartite graph is one where all possible edges between vertices of V_1 and V_2 are in E .

Given any metric space on a finite set of locations V and with distances d_{uv} between locations

$u, v \in V$, we can construct an undirected, weighted graph $G = (V, E)$ where the vertices of G are the same locations as in the metric space and there is an edge of weight d_{uv} between every pair of vertices $u, v \in V$. Similarly, any undirected graph with non-negative edge weights $G = (V, E)$ corresponds to a metric space where the locations are the vertices V and the distance between u and v in the metric space is the length of the shortest path between u and v in G . The connection between weighted, undirected graphs and metric spaces is strong enough that we take the liberty of being able to perform graph operations on a metric space. Note that we will assume that all graphs have non-negative edge weights.

An optimization problem is a problem in which each instance I has a set of feasible solutions F_I and an objective function $M_I : F_I \rightarrow \mathbb{R}$ on feasible solutions where the goal is to find some $f \in F_I$ that minimizes or maximizes $M_I(f)$. Only minimization problems are considered in this thesis so all future discussion on optimization problems will assume the goal is find a solution with minimum cost. The optimal cost for an instance I is defined as $\min_{f \in F_I} M_I(f)$ and an optimal solution is any $f \in F_I$ that achieves this minimum value. An NP-hard optimization problem is a problem whose decision version is NP-hard. That is, given an instance I of the problem and a value k , it is NP-hard to determine if there exists $f \in F_I$ such that $M_I(f) \leq k$.

An α -approximation algorithm for an optimization problem is an algorithm which produces a feasible solution for any instance of the problem that is within a factor α of the optimal solution. In other words, if OPT is the cost of an optimal solution for a given instance of the minimization problem, then an α -approximation finds a feasible solution of cost T where $T \leq \alpha \cdot OPT$. Unless stated otherwise, all approximation algorithms discussed or referenced in this thesis run in polynomial time. A *polynomial-time approximation scheme* is an approximation algorithm which, for any fixed constant $\epsilon > 0$, finds, in polynomial time, a solution of cost T where $T \leq (1 + \epsilon) \cdot OPT$ (though the running-time may be exponential in $\frac{1}{\epsilon}$). Finally, an approximation algorithm with an *additive* ratio of α means that the solution found by the algorithm has cost at most $OPT + \alpha$.

An optimization problem $\mathcal{P}$ is in the class APX if there is a polynomial-time approximation algorithm that finds a solution within a constant factor of the optimal solution. We say that a problem $\mathcal{P}$ is APX-hard if for any $\mathcal{R} \in \text{APX}$, there is a polynomial time reduction that transforms an instance R of $\mathcal{R}$ to an instance P of $\mathcal{P}$ such that any $(1 + \alpha\epsilon)$ -approximate solution to P corresponds to a $(1 + \epsilon)$ -approximate solution for R for a fixed α and any $\epsilon > 0$ (a PTAS reduction). In other words, a $(1 + \alpha\epsilon)$ -approximation algorithm for $\mathcal{P}$ yields a $(1 + \epsilon)$ -approximation algorithm for $\mathcal{R}$. There is a problem in APX that is known not to have a β -approximation for some constant β unless $P = NP$ (see, e.g., [18]). Therefore, if a problem $\mathcal{P}$ is APX-hard, then there is no polynomial-time α -approximation algorithm for $\mathcal{P}$ for some constant $\alpha > 1$ (specific to $\mathcal{P}$) unless $P = NP$. This implies a problem that is APX-hard cannot have a PTAS unless $P = NP$. In general, a hardness of approximation result for an NP-hard optimization problem $\mathcal{P}$ is typically a proof that no α -approximation algorithm exists for $\mathcal{P}$ for some α . Many such results rely on an unproven,

but plausible, complexity theory assumption such as $P \neq NP$ or $NP \not\subseteq DTIME(n^{O(\log \log n)})$. Such results are not limited to ruling out polynomial-time approximation algorithms; they may also rule out approximation algorithms with super-polynomial running time (e.g. quasi-polynomial time $O(n^{\text{polylog}(n)})$).

A *linear program* (or LP) is an optimization problem where the goal is to minimize or maximize a linear function based on a set of linear inequalities. A minimization problem expressed as an LP with n variables and m equations takes the form:

$$\begin{aligned} \text{minimize : } & \sum_{j=1}^n c_j x_j \\ \text{subject to : } & \sum_{j=1}^n a_{ij} x_j \geq b_i \quad \forall i = 1, \dots, m \\ & x_j \geq 0, \quad \forall j = 1, \dots, n \end{aligned}$$

where the a_{ij} , c_j and b_i are specified constants. The first expression (the one to be minimized) is called the objective function and the remaining expressions are the constraints. The goal is to find an assignment to the variables x_j such that all constraints are satisfied and the objective function obtains its minimum possible value. An assignment of values to the x_j variables is called *feasible* if all of the constraints are satisfied. Introducing variables y_i for $i = 1, \dots, m$, the *dual* program is defined as:

$$\begin{aligned} \text{maximize : } & \sum_{i=1}^m b_i y_i \\ \text{subject to : } & \sum_{i=1}^m a_{ij} y_i \leq c_j \quad \forall j = 1, \dots, n \\ & y_i \geq 0 \quad \forall i = 1, \dots, m \end{aligned}$$

A maximization linear program and its dual are similarly defined. It is easy to see that the dual of a dual of a linear program is the original program itself. Therefore, we say that a linear program and its dual form a *primal-dual pair* where the primal refers to the original program. One of the most important results in the theory of linear programming is the *LP-duality theorem* which is stated as follows when the primal is a minimization problem (see, e.g., Chvatal [12]).

Theorem 1.2.1 *The primal program has a finite optimum if and only if its dual has a finite optimum. Moreover, for all feasible variable assignments $\bar{x}$ to the primal and feasible variable assignments $\bar{y}$ to the dual we have $\sum_i b_i y_i \leq \sum_j a_j x_j$ with equality if and only if both $\bar{x}$ and $\bar{y}$ are optimal variable assignments for their respective programs.*

An *integer program* (IP) is a linear program that is allowed to only take integer values for the variables. A common technique in combinatorial optimization is to express a problem as an integer

program. For example, the *minimum vertex cover* problem asks to find the smallest collection of vertices of a graph such that each edge is incident to a selected vertex. To model this problem as an IP, introduce $\{0, 1\}$ -valued variables x_v for each vertex $v \in V$ where $x_v = 1$ corresponds to choosing v in the cover and $x_v = 0$ corresponds to leaving v out of the cover. For a given graph $G = (V, E)$, the following integer program models the minimum vertex cover problem in G :

$$\begin{aligned} & \text{minimize} && \sum_{v \in V} x_v \\ & \text{subject to} && x_u + x_v \geq 1 \quad \forall (u, v) \in E \\ & && x_v \in \{0, 1\} \quad \forall v \in V \end{aligned}$$

The *LP-relaxation* of an IP is a linear program that has the same constraints as an integer program with the exception of allowing variables to take any real value ≥ 0 . For example, we could change the $x_v \in \{0, 1\}$ constraint of the previous integer program to $0 \leq x_v \leq 1$ (or $0 \leq x_v$ since it is easily seen that $x_v \leq 1$ in any optimal solution). For a minimization problem, the optimum value of the LP-relaxation of an IP is at most the optimum value of the IP itself (when they exist) since any feasible integer assignments to the variables is also a feasible fractional assignment.

LP-relaxations are very useful in designing approximation algorithms. Say the optimal value of an IP is OPT_i and the optimal value of its LP-relaxation is OPT_f . If a feasible solution of cost T to an instance can be found that is within a factor α of OPT_f , then the algorithm is an α -approximation algorithm since $T \leq \alpha \cdot OPT_f \leq \alpha \cdot OPT_i$. One way to do this is through the notion of *LP rounding*. In this method, a fractional assignment to the x variables is computed in polynomial time so the objective function has value OPT_f under this assignment. From this, x is "rounded" to integer values through some technique while a bound is maintained on the increase of the cost. Other methods that are mentioned, but not used, in this thesis are the *primal-dual* method and the method of *dual fitting*.

We assume that an MFL instance has the following representation. The metric on $|V|$ locations is specified by a $|V| \times |V|$ matrix d where the entry d_{uv} for $u, v \in V$ is the distance between u and v in the metric space. For each $v \in V$, the number of client pebbles and facility pebbles starting at v is specified by two non-negative integers. When an MFL instance is presented on a graph, we simply consider the metric closure of the graph to match our input description.

Some simplifying assumptions for MFL problems can be made. First, we can assume that all clients on location v are sent to the same destination. Since in the TM-MFL and MM-MFL cases, all clients travel to the nearest facility in an optimal solution so ties can be broken in any way. In the CM-MFL case, consider any location v . If v is the destination of a facility, then none of the clients at v will be moved in an optimal solution. If v is not the destination of a facility then all clients at v must be moved so they might as well move to the same location. Furthermore, we allow the notation d_{ij} where $i, j \in V \cup C \cup F$ such that when indexing with a pebble, we actually mean the start location of that pebble. Also, since there are $|V|$ locations, then at most $|V|$ facilities are

Property	Maximum	Total	Cardinality
Connected	$O(\sqrt{m/OPT})$	$O(\min\{n, m\})$	$O(m^\epsilon)$
Hardness of Connected	open	$\Omega(n^{1-\epsilon})$	$\Omega(\log n)$
$s - t$ Connected	$O(\sqrt{m/OPT})$	$O(n)$	exact
Matching	exact	exact	exact
Directed Connectivity	ϵm	open	$O(m^\epsilon)$
Hardness of Directed Connectivity	$\Omega(n^{1-\epsilon})$	open	$\Omega(\log^2 n)$
$\mathbb{R}^2$ Independent Set	$1 + 3^{-1/2}$ additive	open	PTAS
Mobile Facility Location	2	8 (new)	exact (new)
Hardness of Mobile Facility Location	$(2 - \epsilon)$ (new)	APX-hard (new)	N/A

Table 1.1: Known approximation ratios & lower-bounds for movement problems. All results, unless marked as new, are found in [13].

required to cover these locations. However, these facilities could come from any of the $|V|$ locations so we assume that at most $|V|$ facilities reside on the $|V|$ locations by only considering at most $|V|$ facilities from each location and ignoring the rest. Therefore, we can assume that $|F| \leq |V|^2$.

1.3 Previous Work

Previous to our work, the only results for the general movement setting are presented by Demaine *et. al* in [13]. They consider a variety of problems which involve moving pebbles in an undirected graph so that the final configuration of pebbles satisfies some property P . One property is that the subgraph induced by the vertices with pebbles is connected. Another is the property that the induced subgraph contains two specific nodes s and t in the same connected component. The last property they consider for a weighted undirected graph is that there is a perfect matching of pebbles where two pebbles are said to be adjacent if they are on the same vertex or on adjacent vertices. Next, they consider a directed graph and the property that there is a path from any vertex to a specific vertex v in the subgraph induced by all vertices with pebbles. Finally, they consider the property that no two pebbles are within a distance of less than 1 of each other in $\mathbb{R}^2$ where distance is measured by the Euclidean norm. In these problems, they measure the cost of moving the pebbles to obtain the desired property P by either determining the maximum distance any pebble has to travel (*maximum movement*), the total distance travelled by all pebbles (*total movement*) or the total number of pebbles that had to be moved from their start location (*cardinality movement*).

As mentioned before, they briefly considered the mobile facility location variant when discussing open problems and observed a 2-approximation for the maximum movement case. Approximation algorithms and hardness results for all of these movement problems are summarized in Table 1.1 along with our new results. In this table, n is the number of vertices in the graph, m is the number of edges in the graph, and the results depending on ϵ hold for any constant $\epsilon > 0$. To the best of our knowledge, nothing more has been done for the general class of movement problems previous to our work.

A problem related to the mobile facility location problem is the uncapacitated facility location problem, or UFL, which is a classic and well-studied problem in approximation algorithms and operations research. In an instance of UFL, we are given a metric space on a finite set of locations V where some locations are clients $C \subseteq V$ and others are facilities $F \subseteq V$. Each facility $j \in F$ has an associated opening cost f_j . We are required to select a subset $F' \subseteq F$ of facilities to open and then connect each client $i \in C$ to some open facility $\phi(i) \in F'$. The total cost of the solution is the cost of opening all facilities in F' in addition to the sum of the distances of each client i to the facility $\phi(i)$ to which it is connected. In other words, the cost is given by:

$$\sum_{j \in F'} f_j + \sum_{i \in C} d_{i\phi(i)}$$

The first approximation algorithm for UFL was presented by Hochbaum in 1982 [19] with a ratio of $O(\log |C|)$. The algorithm is essentially an implementation of the $O(\log n)$ -approximation algorithm for the classical *set covering* problem by Chvátal [11] and works in the case where the distances do not satisfy the triangle inequality. This is the best possible approximation algorithm for the general case when the distances are not metric unless $P = NP$. This is because there is a simple reduction from set cover to UFL with distances that are not necessarily metric. Through this reduction, the $\Omega(\log n)$ -hardness of set cover by Alon, Mishkovitz, and Safra [2] immediately implies $\Omega(\log |C|)$ -hardness of this general version of UFL. About 15 years later the first constant-ratio approximation algorithm was presented by Shmoys, Tardos and Aardal [36]. Their algorithm is based on LP-rounding and a filtering technique of Lin and Vitter [29, 30] to obtain an approximation ratio of $3.16 \approx 3/(1 - e^{-2})$, which was later improved to a 2.408-approximation by Guha and Khuller [17]. In the same paper, Guha and Khuller also exhibit that there is no 1.463-approximation for UFL unless $NP \subseteq DTIME(n^{O(\log \log n)})$ by a reduction from the $(1 - \epsilon) \ln n$ hardness result for set cover by Feige [15]. This hardness threshold comes from the root of the expression $1 + 2e^{-x} - x$. LP-rounding was again used by Chudak and Shmoys [10] to achieve a ratio of $1.736 \approx 1 + 2/e$. More recently, LP-rounding was employed by Sviridenko [38] in a 1.582-approximation for UFL which is almost the current best ratio for UFL.

The first constant-factor approximation algorithm for UFL that did not rely on the technique of LP-rounding was a local search algorithm with a ratio of $5 + \epsilon$ for any constant $\epsilon > 0$. The technique was first presented as an empirically good heuristic by Kuehn and Hamburger [28] in 1968. In 1998, Korupolu, Plaxton and Rajaraman [27] showed that the local search technique actually yields a 5-approximate solution and that it can be implemented to run in polynomial time (polynomial in n and $\log \frac{1}{\epsilon}$) to produce a $(5 + \epsilon)$ -approximate solution for any $\epsilon > 0$. The analysis was subsequently tightened to prove that the heuristic yields a $(3 + \epsilon)$ -approximation and examples were shown to exhibit that the algorithm can produce a result that is 3 times the optimal solution by Arya *et al.* [3]. The most recent major improvement in the local search technique was presented by Charikar and Guha in [8] which uses an almost linear number of local search steps and has an approximation

ratio of $2.414 \approx 1 + \sqrt{2}$.

Jain and Vazirani [25] took a different approach by using the UFL LP-relaxation to develop a 3-approximation using the primal-dual scheme. Their method has the advantage of being easily adapted to approximate the capacitated facility location problem as well as being used as an auxiliary procedure for approximating the k -median problem. The method of dual fitting has also been applied to UFL through a series of papers [32, 24, 23, 34, 33] and the end result is a quasi-linear-time 1.52-approximation algorithm which is currently the best approximation ratio. In the analysis of this algorithm, a linear program $\mathcal{F}_n$ is described for every integer $n > 0$ called the *factor revealing* LP. It is then shown that the solutions produced by their algorithm are no more than a factor of $z_{|C|}$ greater than the optimal solution where z_n is the optimal objective function value of $\mathcal{F}_n$. Finally, it is proven that $z_n \leq 1.52$ for all $n > 0$ to prove the approximation ratio of the algorithm.

The k -median problem is related to UFL in the sense that it deals with opening facilities and assigning clients to facilities. In this variant, there is no cost associated with opening facilities while we are restricted to opening at most k facilities where k is a specified integer. The first approximation algorithms for the k -median problem were developed by Lin and Vitter [29, 30]. One of their algorithms has a ratio of $(1 + \epsilon)$ while opening at most $(1 + 1/\epsilon)(\ln n + 1)$ facilities and another has a ratio of $2(1 + \epsilon)$ and opens at most $(1 + 1/\epsilon)k$ facilities.

The first approximation algorithm that opened at most k facilities was presented by Bartal in his work on approximating arbitrary metrics by tree metrics [5]. His algorithm would open at most k facilities at an expected cost of at most $O(\log n \log \log n)$ by probabilistically creating a tree based on the given metric and solving the k -median problem exactly on the tree using a simple dynamic-programming procedure. This procedure was improved to a deterministic $O(\log k \log \log k)$ -approximation algorithm by Charikar, Chekuri, Goel and Guha [7]. Subsequent improvements in approximating metrics by tree metrics yield a probabilistic $O(\log n)$ -approximation [14].

A constant approximation for k -median that opened at most k facilities was later presented by Charikar, Guha, Tardos and Shmoys [9]. Using LP-rounding and a filtering technique of Lin and Vitter [29, 30], they were able to construct an integer solution that is at most $6\frac{2}{3}$ times the cost of the optimal fractional solution. As mentioned previously, Jain and Vazirani were able to use their primal-dual 3-approximation for UFL as an auxiliary function to obtain a 6-approximation for the k -median problem through the technique of Lagrangian relaxation [25]. Charikar and Guha [8] improved this technique to produce a 4-approximation.

The best approximation algorithm to date for the k -median problem is a local search technique by Arya *et. al* [3]. Their algorithm begins by selecting an arbitrary set of k locations to be the facilities. The greedy heuristic of simultaneously closing one facility and opening another is used as long as the cost can improve in this way. When a local minimum is reached through this process, they prove that its cost is at most 5 times the cost of the optimal solution. They then generalize this

method to simultaneously open and close at most p facilities in any step for some constant p and show that a local minimum has cost at most $3 + 2/p$ times the optimal solution. They also show how the search can be implemented to run in polynomial time to get a $(3 + \epsilon)$ -approximation for any constant $\epsilon > 0$. For the hardness of approximating k -median, Jain, Mahdian and Saberi [24] prove that there is no approximation algorithm with a ratio of $1 + 2/e - \epsilon \approx 1.735 - \epsilon$ for any $\epsilon > 0$ unless $NP \subseteq DTIME(n^{O(\log \log n)})$ using the same set cover hardness result by Feige [15] used to prove the hardness of UFL.

A variant of k -median is the k -center problem which asks to open k facilities to minimize the maximum distance any client must travel to its nearest open facility. This can be viewed as the min/max variant of k -median. Hochbaum and Shmoys developed a combinatorial 2-approximation algorithm [20] while Hsu and Nemhauser proved that there is no α -approximation for the k -center problem for any $1 \leq \alpha < 2$ unless $P = NP$.

These problems all have strong similarities to the problems we consider in this thesis. In fact, one of our hardness proofs for TM-MFL will use very simple reduction from an APX-hard subclass of UFL. However, one could argue that TM-MFL is more closely related to the k -median problem since we are dealing with a restricted number of facilities. In fact, our constant TM-MFL approximation algorithm is motivated by the first constant-ratio approximation algorithm for k -median in [9]. Moreover, we get a better hardness result (modulo a stronger complexity assumption) from the known hardness of k -median. The k -center problem corresponds to MM-MFL in the same manner that k -median corresponds to TM-MFL. Moreover, we already have a 2-approximation for MM-MFL from Demaine *et. al* [13] and we also prove that there is no better than a 2-approximation for MM-MFL unless $P = NP$.

1.4 New Results

A few preliminary results regarding MFL are presented in Chapter 2. We begin by showing that CM-MFL can essentially be solved by sorting after some simple pre- and post-processing steps. Following this, the 2-approximation for MM-MFL that was observed in [13] is presented with a proof of the ratio. The chapter is concluded by presenting a pseudo-polynomial-time dynamic programming algorithm for solving TM-MFL when the underlying metric is a tree where the running time will be polynomial in the number of locations and the number of client pebbles. This result can be turned into a pseudo-polynomial-time $O(\log n)$ -approximation for the case of general metrics.

Chapter 3 begins by addressing the problem of improving the 2-approximation for MM-MFL by proving that it is NP-hard to determine if an MM-MFL instance has a solution of cost at most 1 or if all solutions have cost at least 2. This shows that there cannot be any α -approximation algorithm for $1 \leq \alpha < 2$ for MM-MFL unless $P = NP$. Following this, APX-hardness of TM-MFL is proven by a simple reduction from a subclass of the classic uncapacitated facility location problem that is known to be APX-hard. In addition to APX-hardness, we also show that TM-MFL essentially

contains the well-studied k -median problem (with a little overhead). That is, we show that any α -approximation algorithm for TM-MFL immediately implies an $(\alpha + o(1))$ -approximation algorithm for k -median.

Chapter 4 presents the main result of this thesis; a constant-factor approximation algorithm for TM-MFL that is based on LP-rounding. The presentation appears in stages and begins by outlining a randomized algorithm with an expected ratio of 16. The algorithm first computes the optimal fractional solution to the LP relaxation to the natural IP formulation of the problem. From this, the solution to the LP undergoes a series of modifications (one of which is a randomized procedure) to bring the variable assignments closer to integers while bounding the cost increase. The result is an integer solution that is bound from the optimal fractional solution by an expected factor of 16. Following this, a few improvements to the analysis and execution of the algorithm are presented to improve its complexity. First, a slight modification is made to one of the steps to improve the bound on the approximation ratio. Then, the analysis of the expected cost increase of the last step is tightened to improve the approximation guarantee. Finally, the standard method of conditional probabilities is applied to the randomized rounding step to deterministically compute a feasible answer that has cost at most the expected cost of the randomly generated answer. The end result is a deterministic algorithm that computes, in polynomial time, a solution that has cost at most 8 times the cost of the optimal fractional solution. To conclude this section, a deterministic 5-approximation is observed when we allow some of the constraints of the IP formulation to be violated by a constant factor.

Chapter 5 summarizes the work of this thesis and gives our concluding remarks on the problems we explored. Open problems are presented which can be used to direct future work in this area. All new results appearing in this thesis were developed during the author's graduate studies. The hardness results of Chapter 3 as well as the deterministic 8-approximation of Chapter 4 appear in [16].

Chapter 2

Initial Results

2.1 Solving CM-MFL

Recall that the CM-MFL variant of the mobile facility location problem asks to move the clients and facilities to a feasible solution while minimizing the number of pebbles that were moved from their start location. First, we present the solution for the case when no two facilities start on the same location. As discussed in Section 1.2, we can assume that all clients starting from a common location v have the same destination. Moreover, it is easy to see that we can assume that all pebbles (both clients and facilities) starting at a common location v have the same destination. For each $v \in V$, say P_v is the number of pebbles (both client and facility) that start at location v . In the optimal solution, we have that $P_v \leq P_{v'}$ for all locations $v, v' \in V$ such that v is not the destination of a facility and v' is. If there was such a pair of locations with $P_v > P_{v'}$, then we could move all pebbles whose destination is v' to v . Since v now has a facility, then we are no longer required to assign the P_v clients to another location. In this way, we save moving P_v clients and must move $P_{v'}$ clients that did not previously move for a net decrease in the overall number of pebbles moved. Furthermore, we can assume that each location v is the destination of at most one facility since we could move facilities from v back to their start locations until only one facility remains. Given that all facilities have different start locations, then this reassignment is always possible. If we order the locations $v_1, v_2, \dots, v_{|V|}$ so that $P_{v_i} \geq P_{v_{i+1}}$ for all $1 \leq i < |V|$ while breaking ties by saying if v_i is not the destination for some facility then v_{i+1} is not the destination for some facility as well, then we see that the cost is $|C| + |F| - \sum_{i=1}^{|F|} P_{v_i}$. This cost is optimal since at most $|F|$ locations are not required to reassign their pebbles in any solution and the quantity expressed above has the $|F|$ most populated locations not moving their pebbles.

We claim that we can easily find a solution that moves the minimum number of pebbles by executing the procedure in Figure 2.1. Since we choose $|F|$ locations as facility destinations and since no location in the final solution receives more than one facility, then the procedure produces a feasible solution. Also, the cost of the solution produced by the procedure is exactly $|C| + |F| - \sum_{i=1}^{|F|} P_{v_i}$. It is then easy to see that this is the same cost as the optimum solution.

```

sort the  $v \in V$  so that  $P_{v_1} \geq P_{v_2} \geq \dots \geq P_{v_{|V|}}$ 
let  $U = \{v_1, v_2, \dots, v_{|F|}\}$ 
for  $i = 1 \dots |V|$  do
    if  $i \leq |F|$  then do not move the pebbles from  $v_i$ .
    else if  $v_i$  is the start location for some facility  $f$ 
        then move pebbles at  $v_i$  to some  $u \in U$  that is not yet assigned a facility
    else assign the pebbles at  $v_i$  to any location in  $U$ 

```

Figure 2.1: Solving CM-MFL when no two facilities start at the same location.

With this as an auxiliary function, we have the following result.

Theorem 2.1.1 *There is a simple, polynomial-time exact algorithm for CM-MFL.*

Proof. Recall that we can assume, without loss of generality, that $|F| \leq |V|^2$. Modify the instance so that no two facility pebbles start at the same location by creating dummy locations v_f for a facility f starting on v and moving the start location of f to v_f until only one facility remains on location v . Any feasible solution to the new instance corresponds to a feasible solution of at most the same cost in the original instance by simply moving all assignments at dummy locations v_f to the original location of the facility f . Any client that was being assigned to a dummy location v_f is now assigned to the original location of facility f and the remaining client assignments do not change so at most the same number of clients must move. Any facility f' that was being assigned to a dummy location v_f is now being assigned to the start location of f . If $f' = f$, then this facility does not move in either solution and if $f' \neq f$, then this facility must have moved in the solution to the modified instance. Therefore, the number of facilities that move does not increase.

Similarly, any solution to the original instance corresponds to a solution in the modified instance of at most the same cost. Consider each location $u \in V$ that receives a facility in a given solution to the original instance. If u is not the start location of a facility, then assign the same clients and facilities to u in the new instance as in the original instance. The same number of clients and facilities move from their original start locations to u in both instances. Now, if u is the start location of some facility in the original instance then we can assume that there is a facility f that starts at u which does not move. Otherwise, if all facilities starting at u are assigned elsewhere, then there must be some facility f' starting at a location $u' \neq u$ such that f' is assigned to u (since u is assigned some clients). Choose any f starting at u and say that u'' is its destination. We can, instead, assign f' to u'' and f to u which strictly decreases the number of facilities that move. Now, for each u that is the destination of some clients and the start location of a facility f , we can assume, without loss of generality, that the start location of f is not moved to a dummy node. If we keep this assignment of client destinations to u in the modified instance while not moving f , then we have a feasible solution to the modified instance of at most the same cost.

Therefore, since both the original and modified instances have the same optimal solution to CM-

MFL and no two facilities start at the same node in the modified instance, we can use the algorithm in Figure 2.1 to find the optimal solution for any CM-MFL instance. $\square$

2.2 A 2-Approximation for MM-MFL

Demaine *et. al.* [13] observed, without proof, a simple 2-approximation for minimizing the maximum distance any pebble moves for the mobile facility location problem (MM-MFL). For the sake of completeness, we present the algorithm with proof in this thesis. This result will be complemented by proving (in Chapter 3) there is no approximation algorithm with a ratio better than 2 for MM-MFL unless $P = NP$.

Theorem 2.2.1 [13] *There is a polynomial-time 2-approximation for MM-MFL.*

Proof. The algorithm is very basic. Do not move any facility pebbles from their start locations and move each client pebble to its nearest facility pebble. Say the cost of this solution is T for a given instance of MM-MFL. Consider a client pebble c that moves to a facility pebble f such that $T = d_{cf}$. Say also that in an optimal solution, client c moves to location v which is assigned facility f' . We then bound T as follows:

$$\begin{aligned} T &= d_{cf} \\ &\leq d_{cf'} \\ &\leq d_{cv} + d_{vf'} \\ &\leq OPT + OPT \end{aligned}$$

The first inequality follows from c being moved to f instead of f' , the second from the triangle inequality and the third from the fact that c and f' both move to v in the optimal solution. $\square$

2.3 A Pseudo-Polynomial-Time Exact Algorithm for TM-MFL in Trees

The TM-MFL problem asks to find a feasible solution to the mobile facility location problem while minimizing the total distance moved by all pebbles. Consider an instance of TM-MFL where each location is a vertex in a weighted tree and the distances d_{uv} between locations u and v is defined as the length of the unique path from u to v . Rather than computing the metric closure of this graph, we instead keep an adjacency list so that we can utilize properties that are particular to a tree.

Fix any vertex r to be the root of this tree. From this, ensure that each vertex has at most two children by the standard method of taking any two children v_1, v_2 of a vertex v with at least three children and creating a new vertex v' . Modify the edges of the tree by attaching v' to v with weight 0, attaching v_1 and v_2 to v' with respective weights $w(v, v_1)$ $w(v, v_2)$ and removing the edges

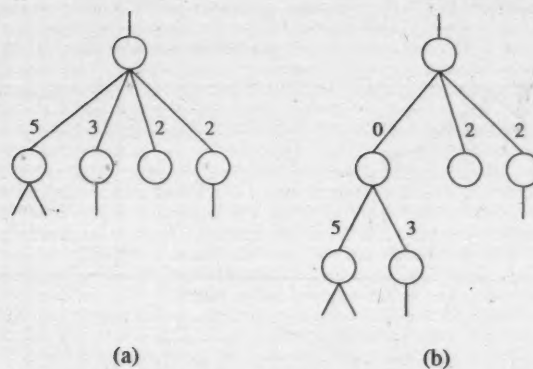


Figure 2.2: One step of fixing a vertex so that it has at most 2 children.

connecting v_1 and v_2 to v . Figure 2.2 shows an example of one step of this process on a node with more than two children. Any solution to the TM-MFL problem in this tree that has pebbles assigned to this new vertex v' can have the pebbles assigned to v instead at no increase in cost. Moreover, each vertex that originally had $d > 2$ children had $d - 2$ new nodes created so there are at most $2|V|$ nodes in the new tree.

The following lemma will simplify the presentation of the algorithm.

Lemma 2.3.1 *There is an optimal solution such that no edge (u, v) has a facility f traveling from u to v and another facility f' traveling from v to u to reach their final destinations. Additionally, we can assume that no two clients traverse an edge in opposite directions.*

Proof. Assume that this were true for a given edge (u, v) . Let f be a facility traveling from u to v with the final destination of v_f and let f' be a facility traveling from v to u with the final destination of $v_{f'}$. We could, instead, change the assignments of f and f' so that f goes to $v_{f'}$ and f' goes to v_f . By moving f to u and then to $v_{f'}$ as well as moving f' to v and then to v_f , we see that every edge, except for the (u, v) edge, is used at most the same number of times. Since the (u, v) edge is no longer used, then this does not increase the total cost. After performing this “uncrossing” step, no edge is traversed by more pebbles than it was before the uncrossing. Moreover, the edge (u, v) that was considered in the uncrossing step is now used for strictly fewer pebble moves. Therefore, repeatedly applying this uncrossing step is a finite process and produces a solution of no greater cost than the original solution. The same arguments hold for why we can additionally assume that no two clients traverse an edge in opposite directions. $\square$

On the other hand, we cannot assume that all pebbles traverse an edge in the same direction. It may be (as in Figure 2.3) that a client and a facility pebble move in opposing directions across an edge.

The main algorithm follows the standard template of dynamic programming in a tree. At each node (except for the root), we will try all possible movements of pebbles to and from the parent node and determine the minimum cost of covering all clients that are left in the subtree rooted at

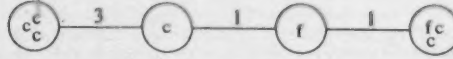


Figure 2.3: The unique optimum solution of cost 6 has a facility at each endpoint which causes a client and a facility to cross paths in the middle.

RecursiveSolve(v, c, f)

let $c' \leftarrow$ the number of clients starting at v
 $f' \leftarrow$ the number of facilities starting at v

if $t[v, c, f]$ has already been computed then return $t[v, c, f]$
else initialize $t[v, c, f] \leftarrow \infty$

if v has no children then

if $c \leq c'$ and $f \leq f'$ and $(c = c' \text{ or } f < f')$ then $t[v, c, f] \leftarrow 0$

else if v has one child v_1 then

for all $(c_1, f_1) \in [-|C|, |C|] \times [-|F|, |F|]$ do

if $c \leq c' + c_1$ and $f \leq f' + f_1$ and $(c = c' + c_1 \text{ or } f < f' + f_1)$ then

$t[v, c, f] \leftarrow \min \{t[v, c, f],$
 $(|c_1| + |f_1|) \cdot w(v, v_1) + \text{RecursiveSolve}(v_1, c_1, f_1)\}$

else if v has two children v_1, v_2 then

for all $(c_1, f_1), (c_2, f_2) \in [-|C|, |C|] \times [-|F|, |F|]$ do

if $c \leq c' + c_1 + c_2$ and $f \leq f' + f_1 + f_2$ and

$(c = c' + c_1 + c_2 \text{ or } f < f' + f_1 + f_2)$ then

$t[v, c, f] \leftarrow \min \{t[v, c, f],$
 $(|c_1| + |f_1|) \cdot w(v, v_1) + \text{RecursiveSolve}(v_1, c_1, f_1) +$
 $(|c_2| + |f_2|) \cdot w(v, v_2) + \text{RecursiveSolve}(v_2, c_2, f_2)\}$

return $t[v, c, f]$

Figure 2.4: Pseudo-polynomial-time algorithm for solving TM-MFL in a tree.

the node in consideration. This will be recursively computed in a post-order traversal of the tree in that we determine the minimum cost of the sub-problems rooted at the children of a given node v before computing the value of the sub-problem rooted at v . For each $v \in V, c \in [-|C|, |C|]$ and $f \in [-|F|, |F|]$ we compute minimum cost of properly assigning all clients and facilities to nodes in the subtree rooted at v when c clients and f facilities travel from v to its parent (negative numbers denote receiving pebbles from the parent). The cost of this sub-problem will be computed as $t[v, f, c]$ where the cost of moving $|c|$ clients and $|f|$ facilities between v and its parent does not contribute to $t[v, f, c]$. We use the symbol ∞ and treat it in the only reasonable way for this problem as $\infty + x = \infty$ for any $x \in \mathbb{Z} \cup \{\infty\}$, $x < \infty$ for $x \in \mathbb{Z}$ and $t[v, c, f] = \infty$ if and only if there is no feasible solution for the corresponding sub-problem. The details of the algorithm are presented in Figure 2.4 where we initially call RecursiveSolve($r, 0, 0$) to solve the TM-MFL problem in this tree with root r .

Theorem 2.3.2 *The algorithm in Figure 2.4 computes the cost of the optimal solution for a TM-MFL*

instance in a tree with a running time that is polynomial in $|V|$, $|C|$ and $|F|$.

Proof. We prove that for each $v \in V$, $c \in [-|C|, |C|]$ and $f \in [-|F|, |F|]$ that $t[v, c, f]$ is the minimum cost of serving clients and facilities in the subtree rooted at v when c clients and f facilities are moved from v to its parent (negative c or f mean that v receives pebbles from the parent) by discussing each case while remembering that $t[v, c, f] = \infty$ if this is not possible.

For the base case of a node v with no children, there will be exactly $c' - c$ clients and $f' - f$ facilities remaining after the move. The conditions ensure that there are enough clients and facilities starting at v and if a client is left at v then a facility must be left as well. Since v has no children, then the pebbles that are not moved up to its parent can be covered with cost 0.

Similarly, if v has one child v_1 , then we try all possible ways of moving pebbles across edge (v, v_1) . Since c clients are sent from v to its parent, c' clients start at v , and c_1 pebbles are sent from v_1 to v , then we see that the first condition ensures that there are enough client pebbles to perform this movement. Similarly, the second condition ensures that there are enough facility pebbles to perform this move. Finally, the last condition ensures that any client that is not moved is covered by a facility. Thus, the second value in the min expression is the minimum cost of covering all clients in the subtree rooted at v when c clients and f facilities are moved up to the parent and c_1 clients and f_1 facilities are moved from v_1 to v . Since all possibilities for c_1 and f_1 are considered, then this minimum value for $t[v, c, f]$ is correctly computed. Note that if it is not possible to cover all clients in the subtree rooted at v_1 when c_1 clients and f_1 facilities are moved there, then the second value in the min expression will be ∞ .

The third case of v having two children is argued in a similar fashion to the previous case. Therefore, since the root r will have no clients or facilities traveling to its (non-existent) parent, the value $t[r, 0, 0]$ will be correctly computed to be the minimum cost of covering all clients when $\text{RecursiveSolve}(r, 0, 0)$ is called.

Since each $t[v, c, f]$ value is computed exactly once and since $(c, f) \in [-|C|, |C|] \times [-|F|, |F|]$, then $O(|V||C||F|)$ such values must be computed. In the worst case, when v has two children then $O(|C|^2|F|^2)$ executions of the loop body are made of which each takes constant time (except for the recursive call). Therefore, each $t[v, c, f]$ value has $O(|C|^2|F|^2)$ work done in its computation so the overall the running time is $O(|V||C|^3|F|^3)$. $\square$

If, in addition to the cost of the optimal solution, the final configuration of the pebbles is desired, we simply add the overhead of remembering the arg min of each min value that is computed to get the number of times each edge is crossed by facilities in the optimal solution determined by the algorithm (we do not need to keep track of clients). That is, for each edge, we know the value $f(u, v) \in [-|F|, |F|]$ which denotes the number of facility pebbles that travel from u to v (or v to u if the number is negative). For each location v , let f_v denote the number of facility pebbles starting at v . The number of facility pebbles whose final destination is v is then $f_v + \sum_u f(u, v)$ where the

sum is taken over all vertices u adjacent to v . We then discuss how to compute an optimal solution if, for each location v , we are given how many facilities have v as their destination.

First, we determine the optimal assignment for facilities by finding a minimum weight perfect matching in a bipartite graph. For each facility j , create a vertex on one side of the bipartition. For each location v , add k_v vertices to the other side of the bipartition where k_v is the number of facilities whose destination is v . Connect a vertex representing facility j to a vertex representing location v with weight d_{jv} . We can compute a minimum weight perfect matching in this bipartite graph in $O(|F|^3)$ time (see, e.g. [35] for details). Then for each location v that is the start location of a client, it suffices to move all clients at that location to the nearest location that is the destination of a facility. In a tree, this can be accomplished in linear time per location for a total running time of $O(\min\{|V|^2, |F|^3\})$. The cost of assigning facilities to the locations was minimized as was the cost of moving each client to its nearest facility destination. This solution is feasible and, thus, optimal.

A few minor improvements to the algorithm and its analysis can be made. First, recall that we can assume that $|F| \leq |V|^2$ by having at most $|V|$ facilities start at each node. Moreover, even though we have $|F| \leq |V|^2$, we note that at most $|V|$ facilities must move since there are at most $|V|$ vertices that need to be covered, so we can assume that each (c_i, f_i) pair that is iterated over in the algorithm in Figure 2.4 has $|f_i| \leq \min\{|F|, |V|\} \leq |V|$. These observations imply that the algorithm in Figure 2.4 can be executed in pseudo-polynomial-time $O(|V|^4|C|^3)$. If $|C|$ is bound by a polynomial in the number of nodes of the graph (i.e. $|C| \leq |V|^k$), then we have a polynomial running time of $O(|V|^{4+3k})$. Consider an assignment of facilities in an optimal solution. If facility j is assigned to the start location of facility j' and facility j' is assigned to location v , we can assign facility j to v and facility j' does not need to move. Since we are in a metric space, then this does not increase the cost of the solution. The point is that if we know, for each location v , how many facilities will have v as their start location, then we can greedily assign facilities j to their start locations until there are no more such assignments. Following this, there should only be at most $|V|$ facilities that still need to be assigned, so the matching of facilities to locations can be performed with only at most $|V|$ vertices on either side of the auxiliary bipartite graph. Thus, explicitly constructing a solution as described above can be done in $O(|V|^3)$ time. Finally, we note that the creation of a tree with at most 2 children per node can be regarded as purely a conceptual tool and that the algorithm in Figure 2.4 can be easily adapted to avoid explicitly computing this transformation.

This result implies a pseudo-polynomial time $O(\log n)$ -approximation algorithm for a general metric space. To show this, we describe the method of probabilistically embedding a metric into a tree metric. Consider a metric space M on n locations V with distances d_{uv} between any two $u, v \in V$. Consider another metric space M' on the same set of locations V with distances d'_{uv} . The metric M' is said to dominate the metric M if $d_{uv} \leq d'_{uv}$ for all $u, v \in V$. Finally, let $\mathcal{M}$ be a family of metrics on locations V and let $\mathcal{D}$ be a probability distribution over $\mathcal{M}$. The pair $(\mathcal{M}, \mathcal{D})$ is said to α -probabilistically approximate the metric M if every $M' \in \mathcal{M}$ dominates M and for every two

locations $u, v \in V$, the expected distance between u and v in a metric M' taken from $\mathcal{M}$ according to distribution $\mathcal{D}$ is at most $\alpha \cdot d_{uv}$. We say that α is the distortion of the probabilistic embedding.

Alon, Karp, Peleg and West [1] discuss how to approximate an arbitrary graph metric by a distribution of spanning trees of the original graph with a $2^{O(\sqrt{\log n \log \log n})}$ -probabilistic approximation. Bartal [4] then describes how to probabilistically embed an arbitrary metric into a tree metric with distortion $O(\log^2 n)$ if the tree does not necessarily need to be a spanning tree of the original graph metric. Later, Bartal [5] improved this result to $O(\log \log \log n)$ and exhibited a lower bound of $\Omega(\log n)$ on the distortion of approximating arbitrary metrics by a tree metric. As a final step, Fakcharoenphol, Rao, and Talwar [14] describe how an arbitrary metric space can be approximated by a distribution over dominating tree metrics with distortion $O(\log n)$ as well as how to efficiently sample from this distribution.

To apply these ideas to TM-MFL, start by sampling a tree metric T with respect to the distribution of the set of dominating tree metrics with distortion $O(\log n)$ described in [14]. Have the clients and facilities start at the same location in T as in the original metric and use the pseudo-polynomial time algorithms in Figure 2.4 and its following discussion to compute an optimal answer in T . From this optimal assignment in T , use the same assignment in the original problem to get a solution of cost at most $O(\log n)$ times the optimal solution. This is a pseudo-polynomial time randomized $O(\log n)$ -approximation algorithm for TM-MFL in arbitrary metrics.

Chapter 3

Hardness Results

3.1 MM-MFL is $(2 - \epsilon)$ -Hard to Approximate

The following theorem expresses that the simple 2-approximation to MM-MFL in [13] is the best possible polynomial-time approximation algorithm unless $P = NP$. Specifically, a reduction from an NP-complete problem $\mathcal{P}$ to MM-MFL is presented where the “yes” instances of $\mathcal{P}$ produce instances of MM-MFL with optimal cost 1 and “no” instances produce instances of MM-MFL with optimal cost 2.

Theorem 3.1.1 *For any $\alpha < 2$, there is no α -approximation algorithm for MM-MFL unless $P = NP$.*

Proof. NP-completeness of the classic vertex cover problem, proven by Karp [26], is all that is required for this result. The vertex cover problem is similar to the minimum vertex cover problem described in Section 1.2 except that we are given an integer k and asked to simply decide if it is possible to select k vertices to cover all the edges. Given a graph $G = (V, E)$ and an integer k , we construct an instance of minimum maximum-movement facility location on a new graph H as follows. Let the vertex set of H be $V \cup E \cup \{f_1, \dots, f_k\}$ where each f_i is a new node. Add an edge from every f_i to every vertex in V with cost 1 and place a facility in each f_i . For each $v \in V$ and $e \in E$, if v is an endpoint of e in the original graph G then connect v and e in H with an edge having cost 1. Finally, the set of all clients in this new graph is exactly E . See Figure 3.1 for an example of this reduction. Notice that v_1 and v_4 form a vertex cover of size 2 in (a) while moving the facilities to v_1 and v_4 in (b) results in each client being able to connect to a facility by only traversing a single weight 1 edge to either v_1 or v_4 .

If G has a vertex-cover of size k , then we can obtain a solution in H with maximum movement 1 by moving the k facilities to the vertices in V that correspond to some vertex cover of size k . Then each client $e \in E$ is adjacent to some vertex in V with a facility which can be reached with cost 1. Similarly, if there is a solution of with maximum movement of 1 in H , then it is easy to see that all clients must meet facilities at nodes in V which corresponds to a vertex cover of G of size at most

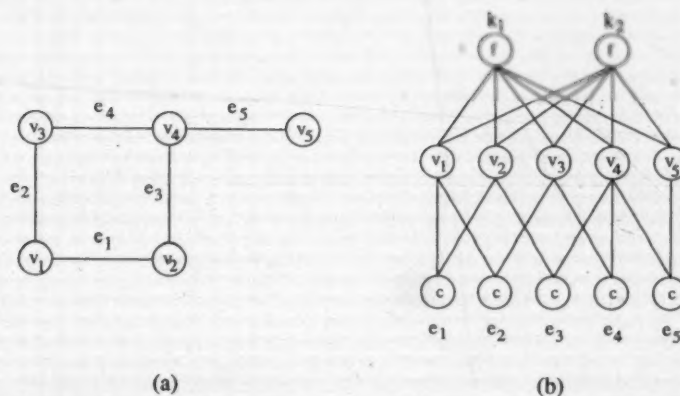


Figure 3.1: Transforming the vertex cover instance in (a) with $k = 2$ to an instance of MM-MFL in (b) where each edge has unit weight.

k . This implies the cost of a solution in H is 2 if G does not have a vertex cover of size k (since we can always move all the clients to a vertex f_i with maximum movement cost being 2).

Consequently, an α -approximation algorithm for any $1 \leq \alpha < 2$ will return a solution of cost less than 2 if G has a vertex cover of size k . Conversely, if G does not have a vertex cover of size k then any algorithm must return a solution of cost at least 2 in H . $\square$

Now, consider the same reduction as in Theorem 3.1.1 except direct all edges towards vertices in V in the newly-created MFL instance. Clients and facilities must meet at locations in V in a feasible solution. In such a solution, it must be that the set of all vertices of V that receive a facility corresponds to a vertex cover of the original graph G . Thus, we get the following corollary:

Corollary 3.1.2 *Determining if there is even a feasible solution to any MFL problem in directed graphs is NP-hard.*

3.2 TM-MFL is APX-Hard

Theorem 3.2.1 *There is some constant α such that there is no α -approximation algorithm for TM-MFL unless $P = NP$.*

Proof. Guha and Khuller [17] prove APX-hardness of the classic (metric) uncapacitated facility location problem UFL. Moreover, their reduction produces an instance where the distances between clients and facilities in the underlying metric are either 1 or 2 and the opening costs of all facilities are some uniform value f . Given such an instance F of the facility location problem with distances $d_{uv} \in \{1, 2\}$ when u is a client and v is a facility, we create a graph H as an instance of TM-MFL. The vertex set of H consists of three disjoint sets U , V , and W . For each client i of F , add a vertex u_i to U . For each facility j of F , add vertex v_j to V and w_j to W and connect v_j to w_j with an edge

of distance f . Connect each pair of vertices $u_i \in U$ and $v_j \in V$ with an edge of distance d_{ij} . Place a client pebble at each node $u_i \in U$ and a facility pebble at each node $w_j \in W$.

For any feasible solution to the mobile facility location problem in H of cost C , we get a feasible solution to UFL in F of at most the same cost. We do this by showing that we can move all facilities that are serving some client to the nodes in V and not increase the cost. If a facility is serving clients in some node $w_j \in W$, then we can move that facility as well as all the clients that it is serving to node v_j . The cost of moving the facility starting at w_j increases by f and the cost of moving the clients assigned to w_j decreases by at least f so the net change in the cost is non-positive. If a facility that was initially located at w_j is moved to some node $u_i \in U$ then move it back to the node v_j along with all clients it is serving. The facility cost decreases by d_{ij} and the cost of connecting client i increases by d_{ij} . All other clients that were being served at u_i were doing so at a cost of at least 2 (shortest path between any two nodes in U has two edges) and each such client can now be served at v_j at a cost of at most 2 resulting in the net change in the cost being non-positive. We can assume that each facility at any location v_j is the facility that was originally located at w_j , since each w_j, v_j edge has distance f (so it is cheaper to move a facility from w_j to v_j instead of from any other $w_{j'}$). So we get a solution to F by opening each facility j in F such that node v_j in H has a facility and connect each client to its nearest facility for a cost at most C .

Similarly, for any feasible solution to the instance F of the facility location problem, we can get a feasible solution to the TM-MFL in H of the same cost by moving each facility corresponding to a opened facility j of F from w_j to v_j and moving each client from u_i to v_j where we have that client i is connected to facility j in the original facility location solution. Thus, the optimum solutions for both problems have the same cost which proves the APX-hardness of minimizing total movement in the mobile facility location problem. $\square$

3.3 Solving the k -Median Problem with TM-MFL

We show how to use an approximation algorithm for TM-MFL to approximate a generalization of the k -median problem. Recall that in the k -median problem, we are given a metric space V and an integer k where some locations $C \subseteq V$ are clients and some locations $F \subseteq V$ are facilities. The goal is to select at most k facilities to open while minimizing the total distance of moving clients to their nearest open facility. Now, consider the generalized version where each client i has an associated integer demand D_i which scales its cost of moving. In other words, the cost of moving client i to facility j is $D_i d_{ij}$. Some of the approximation algorithms for the k -median problem mentioned in Section 1.3 also work when clients have demands.

Theorem 3.3.1 *If there is a polynomial-time α -approximation for TM-MFL for any constant $\alpha \geq 1$, then there is an $(\alpha + o(1))$ -approximation for k -median with integer demands.*

Proof. Suppose we are given an instance of the k -median problem on a metric graph G with an integer k and demands D_i for each client i . First, using scaling, we assume that the minimum distance in G is at least 1. We construct an instance of the mobile facility location problem as follows. Let Δ denote the maximum distance of this metric and define $\sigma = \alpha n k \Delta$, with $n = |V|$. We use the same metric graph G and place k facility pebbles in arbitrary locations of G and place $\lfloor \sigma \rfloor D_v$ client pebbles at each location v . Consider any optimum solution of the instance of TM-MFL with cost $\overline{C} + \overline{F}$, where $\overline{C}$ denotes the cost of moving clients and $\overline{F}$ denotes the cost of moving facilities, and any optimum solution with cost C^* to the k -median instance.

We claim that $\overline{C} + \overline{F} \leq \lfloor \sigma \rfloor C^* + \frac{\sigma}{\alpha n}$. To see this, take the optimum solution of the k -median. Moving the pebbles in TM-MFL as in this solution of k -median costs exactly $\lfloor \sigma \rfloor C^*$. To bring facilities to these k locations costs at most $k\Delta$. Thus we have:

$$\frac{\overline{C} + \overline{F}}{\lfloor \sigma \rfloor} \leq C^* + \frac{k\Delta}{\lfloor \sigma \rfloor} \leq C^* \left(1 + \frac{2}{\alpha n} \right), \quad (3.1)$$

since $C^* \geq 1$ where the last inequality holds for all large enough n .

Now suppose there is an α -approximation algorithm for TM-MFL and it returns a solution with cost $C' + F'$. Obtain a solution to the k -median based on this approximate solution by moving the clients as in the TM-MFL solution, and let C'' be its cost. Using (3.1): $C'' = \frac{C'}{\lfloor \sigma \rfloor} \leq \frac{(C' + F')}{\lfloor \sigma \rfloor} \leq \frac{\alpha(\overline{C} + \overline{F})}{\lfloor \sigma \rfloor} \leq C^* \alpha \left(1 + \frac{2}{\alpha n} \right)$. Thus C'' is within ratio $\alpha + \frac{2}{n}$ of the optimum, i.e. we have an $(\alpha + o(1))$ -approximation for k -median. $\square$

The previous section proves APX-hardness for TM-MFL based on the APX-hardness for UFL. Tracing the constants back through the literature shows that we have only proven a lower-bound of less than 1.1 for the approximability of TM-MFL if $P \neq NP$. Jain, Mahdian and Saberi [24] proved that there cannot be an approximation algorithm with ratio $1 + 2/e - \epsilon \approx 1.735$ for any constant $\epsilon > 0$ unless $NP \in DTIME(n^{O(\log \log n)})$. By Theorem 3.3.1 and the hardness for k -median, we have the same hardness result for TM-MFL.

Chapter 4

An 8-Approximation Algorithm for TM-MFL

4.1 Overview

For ease of exposition, we begin by presenting a randomized 16-approximation algorithm for the minimum total-movement mobile facility location problem. This algorithm uses randomized rounding of the optimal fractional solution obtained from solving the LP-relaxation from a natural IP formulation of the problem. Following this, the presentation and analysis of the algorithm is improved to exhibit an expected approximation ratio of 8. As a final step, we observe that the method of conditional probabilities can be used to derandomize the algorithm to obtain a solution that costs no more than the expected cost of the randomized solution. The last two sections discuss the running time of this algorithm and how the ratio can be further improved if a slight violation of the constraints of the IP formulation is allowed.

Recall that in TM-MFL, we have a metric graph $G = (V, E)$ with costs d_{ij} on the edges, a set of client pebbles C and a set of facility pebbles F . As stated in Section 1.2, we can assume that each location starts with at most $|V|$ facilities so $|F| \leq |V|^2$. Furthermore, we then assume (after the previous observation) that we have each facility starting on a node with no other pebbles; this can be enforced by transforming the general problem by creating a dummy-node for each facility and connecting it to its original node with cost 0. Given this, we can think of F as being a subset of V . Say there are D_v clients on a location v . Since clients starting from a common location can be assigned to the same destination in an optimal solution, then we consider each location v with $D_v > 0$ as a single client with demand D_v and say $C \subseteq V$ is the set of location with non-zero demand. The cost of moving a client at v to another location u is $D_v \cdot d_{vu}$.

Our starting point is an integer programming formulation of the problem. Define indicator variables x_{iv} for each $i \in C$ and $v \in V$, and y_{jv} for each $j \in F$ and $v \in V$; variables x_{iv} and y_{jv} will be 1 if client i or facility j is moved to location v , respectively, and 0 otherwise. Then the goal is to optimize to following program:

$$\begin{aligned}
\text{minimize : } & \sum_{i \in C} \sum_{v \in V} x_{iv} D_i d_{iv} + \sum_{j \in F} \sum_{v \in V} y_{jv} d_{jv} \\
\text{such that : } & \sum_{v \in V} x_{iv} = 1 \quad \forall i \in C \\
& \sum_{v \in V} y_{jv} = 1 \quad \forall j \in F \\
& \sum_{j \in F} y_{jv} \geq x_{iv} \quad \forall i \in C, v \in V \\
& x_{iv} \in \{0, 1\} \quad \forall i \in C, v \in V \\
& y_{jv} \in \{0, 1\} \quad \forall j \in F, v \in V
\end{aligned}$$

The first two constraints ensure that a client or facility has a unique destination while the third ensures that any location that is the destination of some client is also the destination of some facility. We say a location v is covered if there is at least one facility assigned to it (so we can move any client to be served at v). We obtain a linear program (LP) relaxation of this problem by relaxing the last two constraints to non-negativity constraints $x_{iv} \geq 0$ and $y_{jv} \geq 0$. Note that the constraints $x_{iv} \leq 1$ and $y_{jv} \leq 1$ are redundant because if any $x_{iv} > 1$ or $y_{jv} > 1$ then the variable assignments are not feasible. The third constraint in this LP ensures, for each client i and location v , that the total facility assignment to this location is at least the amount that client i has assigned to v . Since the size of this LP is polynomial in the size of the input we can compute the optimum fractional solution $(\bar{x}, \bar{y})$ with objective function value OPT_f (using, e.g., Ye's interior point method [39]). For each client i define $\bar{C}_i = \sum_v x_{iv} d_{iv}$ and for each facility j define $\bar{F}_j = \sum_v y_{jv} d_{jv}$. Note that $\bar{C}_i$ and $\bar{F}_j$ are, respectively, the total costs of moving a unit of demand of a client i and moving a facility j in the optimal fractional solution $(\bar{x}, \bar{y})$. Denote $\sum_i \bar{C}_i D_i$ by $\bar{C}$ and $\sum_j \bar{F}_j$ by $\bar{F}$; so $\bar{C} + \bar{F} = OPT_f$.

Our randomized algorithm produces an integral solution of (expected) cost at most $16\bar{C} + 4\bar{F} \leq 16OPT_f$. The algorithm has five phases, starting from the optimal fractional solution $(\bar{x}, \bar{y})$, where each phase brings the current solution closer to an integer solution while keeping a bound on the cost increases. We begin with a summary of each phase. Since the values of $(\bar{x}, \bar{y})$ change frequently throughout the algorithm, we adopt the following notation. For each step p , $(\bar{x}^{(p)}, \bar{y}^{(p)})$ will denote the assignments of clients and facilities to locations after step p . Similarly, we will let $\bar{C}_i^{(p)}$ and $\bar{F}_j^{(p)}$ denote the costs of moving one unit of demand of client i and moving facility j under the assignment $\bar{x}^{(p)}$ and $\bar{y}^{(p)}$, respectively. Finally, we let $\bar{C}^{(p)} = \sum_i \bar{C}_i^{(p)} D_i$ and $\bar{F}^{(p)} = \sum_j \bar{F}_j^{(p)}$.

Step 1: Clustering of clients: To start, we will create a modified instance of the original problem by moving demands between clients and removing clients with zero demand so that different locations with non-zero client demands are far apart. More specifically, for all pairs of clients $i \neq i'$ we want to make sure that $d_{ii'} > 4 \cdot \max\{\bar{C}_i, \bar{C}_{i'}\}$. This will be guided by the values of $(\bar{x}, \bar{y})$ so that the cost of the new instance under the assignments of $(\bar{x}, \bar{y})$ is at most OPT_f and so we can

recover an integer solution to the original problem from an integer solution to the modified problem by paying only a constant factor of $4\bar{C}$.

Step 2: Relocation of facilities: The next step is to ensure that each location v with $x_{iv}^{(2)} > 0$ for some $i \in C$ is the initial location of some client i' . That is, at the end of Step 2, $x_{iv}^{(2)} > 0$ implies v is the starting location of some client i' . Based on the previous step of clustering the demands and on how we perform this step, we will now be able to say that $x_{ii}^{(2)} \geq \frac{1}{2}$, so less than half of each client i must be served at a location v different from the initial location of client i and that this location v is the initial location of some other client. Moreover, each client i has this amount served at the nearest client i' to i while breaking ties by choosing the client i' with lowest index. Let this closest client to i be denoted by $\phi(i)$. Finally, we remove useless facilities so that each location v now has $\sum_j y_{jv}^{(2)} \leq 1$.

Step 3: Getting a half-integer solution: The third step is more involved. Here we obtain a half-integer solution through a few sub-processes. The first ensures that for each location v , $\sum_j y_{jv}^{(3)} = \frac{a}{2}$ for some integer a by redirecting facility assignments using an augmenting path. The second relies on a matching polytope argument and uses a minimum weighted perfect matching algorithm (in a bipartite graph) to ensure that we can assume each individual $y_{jv}^{(3)} \in \{0, \frac{1}{2}, 1\}$. At this stage, we will have half-integer values for all $x_i^{(3)}$ and $y_j^{(3)}$ variables.

Step 4: Modifying the half-integer solution: We first describe the final step to provide motivation for the fourth step. The final step obtains an integer solution by first fixing a destination for each facility and then assigning each client to a covered location. This step consists of first each facility being uniformly randomly rounded to one of its at most 2 (fractional) destinations. Then we move each client i which is not covered to $\phi(i)$ and from there to $\phi(\phi(i))$ if there is no facility assigned there either, and so on, until the client reaches a covered location. The expected facility cost is exactly the current fractional cost. The expected cost of moving the clients will be bound by moving each client from i to $\phi(i)$, iteratively.

In order to bound the cost of moving clients in the final step, the fourth step will prepare the current half-integer solution for the randomized rounding step. There are two types of clients that we will consider as "bad". The first type consists of clients i with $x_{ii}^{(4)} = 1$ while having two distinct facilities j and j' such that $y_{ji}^{(4)} = y_{j'i}^{(4)} = \frac{1}{2}$. This is bad because the client currently has cost $\bar{C}_i^{(4)} = 0$ (it is completely covered at its own location), but randomly rounding facilities based on their $y_{jv}^{(4)}$ values may result in location i not being covered; therefore we cannot bound the increase in the cost of that client. The second type of bad client is one with $\phi(\phi(i)) = i$, but both i and $\phi(i)$ have their locations covered with weight $\frac{1}{2}$ by different facilities. This is bad since the event where neither i nor $\phi(i)$ are covered results in client i being moved between i and $\phi(i)$ back and forth as we iteratively move client i based on the ϕ mapping. Notice client i being bad in this case implies $\phi(i)$ is also bad. Through this step, we ensure that there is no bad client (or client pairs) and so we are able to say that every client must be eventually covered as it follows $\phi(i)$ and every

```

sort the clients  $i \in C$  so that  $\bar{C}_1 \leq \bar{C}_2 \leq \dots \leq \bar{C}_{|C|}$ 
for  $i = 1 \dots |C|$  do
    if  $\exists i' < i$  such that  $D_{i'} > 0$  and  $d_{ii'} \leq 4\bar{C}_i$  then
         $D_{i'} \leftarrow D_{i'} + D_i$ 
         $D_i \leftarrow 0$  ( $i$  is no longer a client)

```

Figure 4.1: Algorithm to cluster the demands so all client i balls R_i are disjoint.

client cost increases, in expectation, by a constant factor of its original half-integer cost.

Step 5: Randomized Rounding As described before, we randomly round each facility location to one of its at most 2 (fractional) destination locations with equal probability. Then we move each client i to $\phi(i)$ and from there to $\phi(\phi(i))$ if there is no facility assigned there, and so on, until the client reaches a covered location.

4.2 A Randomized 16-Approximation Algorithm

The details of the steps mentioned in the previous section are covered in this section. Remember that at the end of step p we have $\bar{x}^{(p)}$ and $\bar{y}^{(p)}$ representing the current client and facility assignments, respectively. Each increase in the cost of a solution after a given step will be bound by the current cost of the previous step. Once the entire algorithm is presented we will examine all cost increases to determine the final approximation ratio.

4.2.1 Clustering of Clients

This phase is similar to the first step in the k -median approximation algorithm based on LP-rounding by Charikar *et. al* [9]. Recall that we start from an optimum LP solution and $\bar{C}_i = \sum_v x_{iv} d_{iv}$. We modify the instance in such a way that the current fractional solution is also a feasible solution for the new instance, and given an integer solution for the new instance, we can obtain an integer solution for the original instance with a bounded increase in the cost. Begin by assigning $(\bar{x}^{(1)}, \bar{y}^{(1)}) \leftarrow (\bar{x}, \bar{y})$. We will now modify the current instance of the problem by moving demands between clients. What we want to ensure is that any two clients with non-zero demand are far from each other. This is achieved by executing the procedure in Figure 4.1 (see Figure 4.2 for an example). Note that if a client i has its demand moved to another client i' , then i is not a client in the modified instance. With this in mind, define, for each remaining client i , a set $R_i \subseteq V$ such that $v \in R_i$ if and only if $d_{iv} \leq 2\bar{C}_i^{(1)}$. This is the closed ball of radius $2\bar{C}_i^{(1)}$ centred at location i in the modified instance.

In the following lemmas we point out some properties of this new instance. The first lemma describes how to obtain an integer solution to the original instance from an integer solution to the new instance with an increase of at most $4\bar{C}$ in cost. The second lemma expresses the fact that this new instance does not get worse in its objective function value under the assignments $(\bar{x}, \bar{y})$. The third lemma states that any two clients are far from each other; a property that is used in obtaining

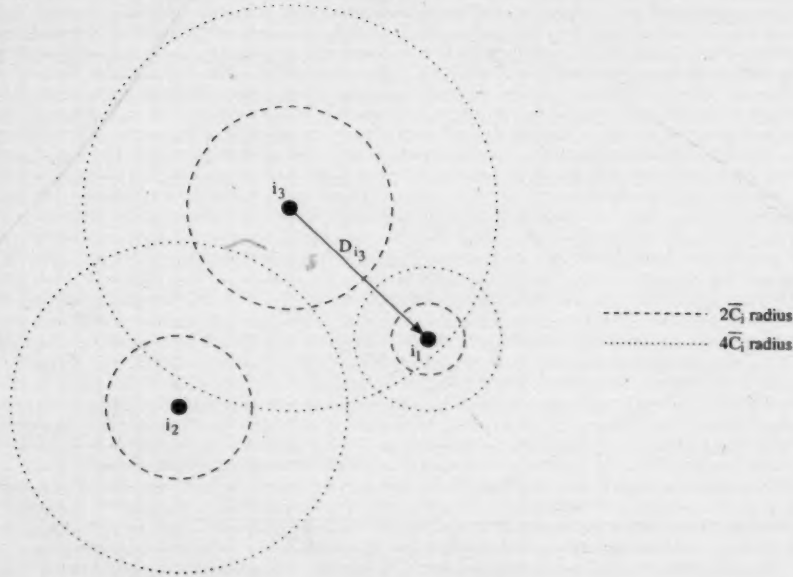


Figure 4.2: Demands are moved from client i_3 to client i_1 since $d_{i_1 i_3} \leq 4\bar{C}_{i_3}$.

an integer solution of the new instance. The final lemma states that the balls R_i are disjoint and each location v in a ball R_i is closer to client i than any other client i' .

Lemma 4.2.1 Any integer solution of cost T to the modified instance can be turned into an integer solution of the original problem with cost at most $T + 4\bar{C}$.

Proof. Simply move the location of each unit of demand in the modified instance to the location of that unit of demand in the original instance. By the definition of the new instance, the cost increase for the unit of demand due to this move is bounded by $4\bar{C}_i$ where i was the original client that possessed that demand. Summing over all units of demand and original clients i , the cost increase is at most $4\bar{C}$. $\square$

Lemma 4.2.2 $\bar{C}^{(1)} \leq \bar{C}$ and $\bar{F}^{(1)} = \bar{F}$.

Proof. Recall that we start with $(\bar{x}^{(1)}, \bar{y}^{(1)}) = (\bar{x}, \bar{y})$. Consider each unit of demand of a client. The contribution of that unit of demand to $\bar{C}^{(1)}$ is exactly the same as in $\bar{C}$ if the demand is not moved. If the demand is moved, then it moves to a new location with a smaller index which implies a reduction in cost since the $\bar{C}_i$ are considered in increasing order. The cost of the facilities stays the same since their assignments were not changed. $\square$

Lemma 4.2.3 Any two clients i and i' in the modified instance have $d_{ii'} > 4 \cdot \max \{ \bar{C}_i^{(1)}, \bar{C}_{i'}^{(1)} \}$.

Proof. Assume, for the sake of contradiction, that there are two clients i and i' such that $d_{ii'} \leq 4 \cdot \max \{ \bar{C}_i^{(1)}, \bar{C}_{i'}^{(1)} \}$. Since i and i' are clients in the modified instance, then they must have been clients in the original instance so $\bar{C}_i^{(1)} = \bar{C}_i$ and $\bar{C}_{i'}^{(1)} = \bar{C}_{i'}$. Assume, without loss of generality, that $\bar{C}_i \leq \bar{C}_{i'}$. When considering client i' while executing the algorithm in Figure 4.1, we have $d_{ii'} \leq 4\bar{C}_{i'}$ so the demand of client i' would be moved to client i so i' is not a client in the modified instance. $\square$

Lemma 4.2.4 For any two distinct clients i and i' in the modified instance, $R_i \cap R_{i'} = \emptyset$. Moreover, for each $v \in R_i$, $d_{iv} < d_{i'v}$.

Proof. Assume, for the sake of contradiction, that there are clients i and i' and a location v such that $v \in R_i \cap R_{i'}$. Then, $d_{vi} \leq 2\bar{C}_i^{(1)}$ and $d_{vi'} \leq 2\bar{C}_{i'}^{(1)}$. By the triangle inequality, $d_{ii'} \leq d_{vi} + d_{vi'}$ so $d_{ii'} \leq 2\bar{C}_i^{(1)} + 2\bar{C}_{i'}^{(1)} \leq 4 \cdot \max \{ \bar{C}_i^{(1)}, \bar{C}_{i'}^{(1)} \}$. This contradicts Lemma 4.2.3.

For the second part, assume that there are clients i and i' such that there is a $v \in R_i$ but $d_{iv} \geq d_{i'v}$. By the triangle inequality, we have $d_{ii'} \leq d_{iv} + d_{i'v}$ which is less than $2d_{iv}$. Since $v \in R_i$, then $2d_{iv} \leq 4\bar{C}_i^{(1)}$ by definition of R_i . Thus, $d_{ii'} \leq 4 \cdot \max \{ \bar{C}_i^{(1)}, \bar{C}_{i'}^{(1)} \}$ which is another contradiction of Lemma 4.2.3. $\square$

From now on, we will be working with the modified instance of the problem.

4.2.2 Relocation of Facilities

First, initialize $(\bar{x}^{(2)}, \bar{y}^{(2)}) \leftarrow (\bar{x}^{(1)}, \bar{y}^{(1)})$. Now, consider each location v with $x_{iv}^{(2)} > 0$ for some client i (i.e. there is some client being served fractionally at v) but there is no client demand located at v . Let $i' \in C$ be any closest client to this location, i.e. $d_{i'v} \leq d_{iv}$ for all other clients $i \in C$. We are going to relocate the clients being served at v to location i' and take the facilities that cover v with them. Let $M = \frac{\max_i x_{iv}^{(2)}}{\sum_j y_{jv}^{(2)}}$ be the fraction of the facility coverage at v that is required to cover all clients i being served at v (it may be that $\sum_j y_{jv}^{(2)} > x_{iv}^{(2)}$ for all $i \in C$, for example when $d_{jv} = 0$ for many facilities j but we will remove all such occurrences after this step). For all clients i and facilities j , perform the following reassignment (see Figure 4.3 for an example):

$$\begin{aligned} x_{ii'}^{(2)} &\leftarrow x_{ii'}^{(2)} + x_{iv}^{(2)} \\ x_{iv}^{(2)} &\leftarrow 0 \\ y_{ji'}^{(2)} &\leftarrow y_{ji'}^{(2)} + M \cdot y_{jv}^{(2)} \\ y_{jv}^{(2)} &\leftarrow (1 - M) \cdot y_{jv}^{(2)} \end{aligned}$$

Lemma 4.2.5 $\bar{C}^{(2)} \leq 2\bar{C}^{(1)}$ and $\bar{F}^{(2)} \leq \bar{F}^{(1)} + \bar{C}^{(1)}$.

Proof. Consider a location v as above where the assignments at v were moved to a nearest client i' . For each client i the cost increases by at most $D_i \cdot x_{iv}^{(1)} \cdot d_{vi'} \leq D_i \cdot x_{iv}^{(1)} \cdot d_{vi}$, where we use the fact

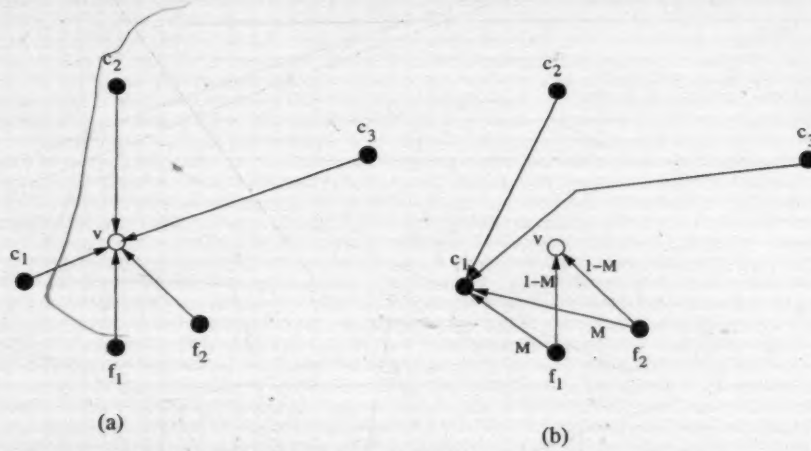


Figure 4.3: Relocating coverage from a location v to its nearest client c_1 . Notice that only an M -fraction of facility assignment is moved to client c_1 from location v .

that $d_{vi'} \leq d_{vi}$ (by definition of i'). This fraction of the client's assignment will not be moved again since it is moved to a client location. Therefore, summing over all i and v , it follows that the cost increase for clients is at most $\bar{C}^{(1)}$.

For each location v we know there is a client i with $x_{iv}^{(1)} = M \cdot \sum_j y_{jv}^{(1)}$. The cost increase incurred by moving facility assignments to v is:

$$M \cdot \sum_{j \in F} y_{jv}^{(1)} (d_{ji'} - d_{jv}) \leq M \cdot \sum_{j \in F} y_{jv}^{(1)} d_{i'v} = d_{i'v} \cdot M \cdot \sum_{j \in F} y_{jv}^{(1)} = x_{iv}^{(1)} \cdot d_{i'v} \leq x_{iv}^{(1)} \cdot d_{iv},$$

which is the fractional cost of moving one unit of demand of client i . The first inequality uses the fact that $d_{ji'} \leq d_{jv} + d_{vi'}$ by the triangle inequality. Notice that when an assignment of a facility is moved from some v to some i' then that fraction of the assignment will never move again in this step. Moreover, each $x_{iv}^{(1)}$ fraction of client assignments will be used at most once in bounding the cost increase of moving facility coverage. Therefore, summing this change in cost over all v shows the cost increase for both clients and facilities is bound by $\bar{C}^{(1)}$. Note that we have used the fact that all demands D_i are integer since they represent the number of client pebbles to be moved. Specifically, we required that $D_i \geq 1$ for all clients. $\square$

Even more can be said about the structure of the current solution.

Lemma 4.2.6 For all clients i : $x_{ii}^{(2)} \geq \frac{1}{2}$. In other words, each client has less than half of its assignment being served at a different location than its own.

Proof. This follows from a simple averaging argument. Consider a client i and the ball R_i . First, we claim that $\sum_{v \in R_i} x_{iv}^{(1)} \geq \frac{1}{2}$, for otherwise, the total cost for fractional values assigned to the vertices outside R_i is strictly larger than $D_i \bar{C}_i^{(1)}$, a contradiction. Since all $v \in R_i$ are strictly closer to client i than to any other client by Lemma 4.2.4, then all assignments $x_{iv}^{(1)}$ for $v \in R_i$ are moved

back to client i 's location. Therefore, at least a total of $\frac{1}{2}$ of the fractional assignment is relocated to the location of client i , or $x_{ii}^{(2)} \geq \frac{1}{2}$. $\square$

Next, for each location v if $\sum_j y_{jv}^{(2)} > 0$ but no client or facility starts at v we do the following (this can happen if at the beginning of this phase, $M < 1$; therefore after relocating the facility values assigned to location v we still have $\sum_j y_{jv}^{(2)} > 0$): for each facility j with $y_{jv}^{(2)} > 0$ we move $y_{jv}^{(2)}$ back to j at no additional cost. Similarly, for each location that is the start location of a client i if $\sum_j y_{ji}^{(2)} > 1$ we can move coverage from facilities from $y_{ji}^{(2)}$ to $y_{jj}^{(2)}$ until $\sum_j y_{ji}^{(2)} = 1$ at no extra cost. Also do this for each location that is the start location of some facility. Since we assume that each facility starts on a location with no other facilities then this is always possible. Thus, in the current solution we have that the only vertices with non-zero coverage are those that are either a client or facility location, $\sum_j y_{ji}^{(2)} \leq 1$ for all locations v and, for all clients i , we have both $x_{ii}^{(2)} \geq \frac{1}{2}$ and $\sum_i x_{ii}^{(2)} = 1$. We can assume that the remaining $1 - x_{ii}^{(2)} \leq \frac{1}{2}$ fraction of each client i not being served at its own location is being served at the nearest client. Denote this client as $\phi(i)$ while breaking ties by the lowest index. Also assume that each client uses the coverage at its own location to the maximum amount. That is, for each i , we can assume that $x_{ii}^{(2)} = \sum_j y_{ji}^{(2)}$ by moving $\sum_j y_{ji}^{(2)} - x_{ii}^{(2)}$ from $x_{i\phi(i)}^{(2)}$ to $x_{ii}^{(2)}$ at no additional cost.

4.2.3 Getting a Half-Integer Solution

In this phase our goal is to ensure that the value of each $x_{iv}^{(3)}$ and $y_{jv}^{(3)}$ is in $\{0, \frac{1}{2}, 1\}$. Start with $(\bar{x}^{(3)}, \bar{y}^{(3)}) \leftarrow (\bar{x}^{(2)}, \bar{y}^{(2)})$. We say that a location v is covered half-integrally if $\sum_j y_{jv}^{(3)} \in \{0, \frac{1}{2}, 1\}$. Given the assignment $(\bar{x}^{(3)}, \bar{y}^{(3)})$, we construct a bipartite graph B in the following manner. For each location v create a vertex on one side of the bipartition and for each facility j create a vertex on the other side of the bipartition. We connect v to j in B with weight $y_{jv}^{(3)}$ only if this weight is positive. The edge weights in any connected component of B must sum to an integer since the sum of the weights of the edges incident to any particular facility must be 1. If there is a location v that is not covered half-integrally in the current assignment $(\bar{x}^{(3)}, \bar{y}^{(3)})$, then there must be another location v' in the same connected component as v that is not covered half-integrally in B since the edge weights in that component must sum to an integer.

While there is still a location v that is not covered half-integrally we execute the following procedure. Find a path $v = v_0, j_1, v_1, j_2, v_2, \dots, v_{k-1}, j_k, v_k = v'$ in the bipartite graph B constructed from v (using current $(\bar{x}^{(3)}, \bar{y}^{(3)})$) to some other location v' that is not covered half-integrally. Define $\alpha_0 = d_{i\phi(i)} D_i$ if v is some client i and otherwise say $\alpha_0 = 0$. Similarly define α_k for v' . These α_0, α_k quantities express the cost of serving one unit of demand for the clients at locations corresponding to v and v' if there is no facility coverage at their location.

Let τ be a constant which we will specify after the entire algorithm is presented. Since we could

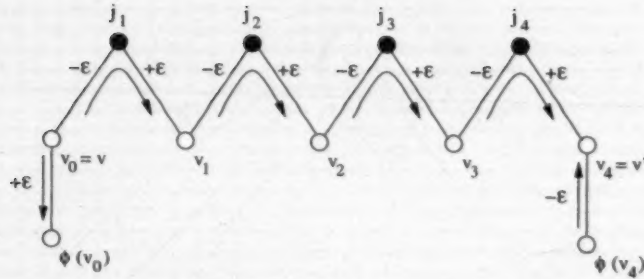


Figure 4.4: Changing fractional assignments over a path between two clients with non-half-integer coverage.

consider this path in the reverse order, without loss of generality, assume that:

$$\alpha_0 + \tau \sum_{m=1}^k d_{j_m v_m} \leq \alpha_k + \tau \sum_{m=1}^k d_{j_m v_{m-1}} \quad (4.1)$$

What we plan to do is shift some coverage from v to v' through this path, by simultaneously increasing each $y_{j_m v_m}^{(3)}$ and decreasing each $y_{j_m v_{m-1}}^{(3)}$ (at the same rate) until one of the edges in the path has value 0 (i.e. the edge disappears from B) or one of the endpoints v or v' is covered half-integrally. Let $\epsilon = \min \{y_{j_m v_{m-1}}^{(3)} \mid 1 \leq m \leq k\}$. If v is a client i , then update $\epsilon \leftarrow \min \{\epsilon, x_{ii}^{(3)} - \frac{1}{2}\}$ which is non-negative by Lemma 4.2.6. Finally, update $\epsilon \leftarrow \min \{\epsilon, 1 - \sum_j y_{jv'}^{(3)}\}$ which is also non-negative since all locations v have $\sum_j y_{jv} \leq 1$ after the previous step. Now, by our construction of B and the assumption that neither v nor v' are covered half-integrally, we have $\epsilon > 0$. Perform the following updates to $(\bar{x}^{(3)}, \bar{y}^{(3)})$:

$$\begin{aligned} y_{j_m v_{m-1}}^{(3)} &\leftarrow y_{j_m v_{m-1}}^{(3)} - \epsilon & \forall 1 \leq m \leq k \\ y_{j_m v_m}^{(3)} &\leftarrow y_{j_m v_m}^{(3)} + \epsilon & \forall 1 \leq m \leq k \\ x_{ii}^{(3)} &\leftarrow x_{ii}^{(3)} - \epsilon & \text{if } v \text{ is a client } i \\ x_{i\phi(i)}^{(3)} &\leftarrow x_{i\phi(i)}^{(3)} + \epsilon & \text{if } v \text{ is a client } i \\ x_{i'j'}^{(3)} &\leftarrow x_{i'j'}^{(3)} + \epsilon & \text{if } v' \text{ is a client } i' \\ x_{i'\phi(i')}^{(3)} &\leftarrow x_{i'\phi(i')}^{(3)} - \epsilon & \text{if } v' \text{ is a client } i' \end{aligned}$$

See Figure 4.4 for an illustration. After each iteration, either an edge of B is deleted (when ϵ is some $y_{j_m v_{m-1}}^{(3)}$ value) or a non-half-integrally covered node is now covered half-integrally (when ϵ is $x_{ii}^{(3)} - \frac{1}{2}$ or $\sum_j y_{jv'}^{(3)}$). Since a new edge in B can never be introduced by this method and all half-integrally covered locations remain so, then there is a polynomial upper-bound on the number of times we must perform this re-assignment of facilities. Denote the difference $\epsilon(\sum_{m=1}^k (d_{j_m v_m} - d_{j_m v_{m-1}}))$ by δ . After a step is performed, the total cost of all clients $\bar{C}^{(3)}$ is changed by $\epsilon(\alpha_0 - \alpha_k)$ since ϵ coverage is removed from v and added to v' . The value of this change in cost is at most $-\tau\delta$ by (4.1). Similarly, the change in the cost of the facilities is δ . Letting Δ be the sum of the δ values over all executions of such an update, we see that $\bar{C}^{(3)} \leq \bar{C}^{(2)} - \tau\Delta$ and $\bar{F}^{(3)} \leq \bar{F}^{(2)} + \Delta$.

The previous process obtains a half-integer solution in that each location v has $\sum_j y_{jv}^{(3)} \in$

$\{0, \frac{1}{2}, 1\}$. However, it is not necessarily true that each $y_{jv}^{(3)}$ is a half-integer. We rectify this situation with a matching.

Lemma 4.2.7 *If there is an assignment $(\bar{x}, \bar{y})$ such $\sum_j y_{jv}$ is a half-integer for all locations v , then we can find an assignment $(\bar{x}', \bar{y}')$ with all y'_{jv} being half-integer where neither the client nor the facility costs increase.*

Proof. We split all locations v with $\sum_j y_{jv} = \frac{a(v)}{2}$ into locations $v_1, \dots, v_{a(v)}$ (note that $a(v) \in \{0, 1, 2\}$). Split each facility j into nodes j_1 and j_2 . Finally, for each original location v with $a(v) > 0$ and each original facility j , set $y_{j_1 v_d} = y_{j_2 v_d} = \frac{y_{jv}}{a(v)}$ for $1 \leq d \leq a(v)$ and keep the original distances. Notice that for each new facility, the sum of fractional values over the edges incident to it is still 1 and each new location is fractionally covered with value exactly 1. This is a fractional matching between the $2|F|$ new facilities and the $2|F|$ new locations of cost $2\bar{F}$ (since we doubled the edges). By total-unimodularity of the matching polytope, we know that there is an integral matching of at most the same cost. Such a matching can be found using a minimum weight perfect matching in bipartite graphs algorithm (e.g. [35]). Consider the new assignment values defined by this matching; all $y_{j_d v_d}$'s are either 0 or 1.

Now in the original problem (before splitting) let $y'_{jv} \leftarrow \frac{1}{2} \sum_{d=1}^{a(v)} (y_{j_1 v_d} + y_{j_2 v_d})$ for each location v and facility j . Since we had an integer matching with the new locations and facilities, we now have that $y'_{jv} \in \{0, \frac{1}{2}, 1\}$. Each new location v_d , for all $1 \leq d \leq a(v)$, was covered with weight 1 in the integer matching so each original location v still has $\sum_j y'_{jv} = \frac{a(v)}{2}$. The weights of the assignments in the integer matching were halved to restore the original problem, so now the current cost of the facilities is at most $\bar{F}$. The client assignments $\bar{x}$ do not change so the client costs do not increase. $\square$

Applying Lemma 4.2.7 to our current assignment, we obtain an assignment to the variables $(\bar{x}^{(3)}, \bar{y}^{(3)})$ in which each $y_{jv}^{(3)}$ is half-integer, for all j, v . Since each client i has $x_{ii}^{(3)} \geq \frac{1}{2}$ and uses its own coverage to the maximum amount, it follows that each $x_{ii}^{(3)} \in \{\frac{1}{2}, 1\}$ and if $x_{ii}^{(3)} = \frac{1}{2}$ then $x_{i\phi(i)}^{(3)} = \frac{1}{2}$.

4.2.4 Modifying the Half-Integer Solution

We construct an auxiliary directed graph H which has a vertex v_i for each client i and a directed edge from v_i to $v_{\phi(i)}$ with weight $d_{i\phi(i)}$. So each vertex in H has outdegree exactly one and by the definition of ϕ , the edge weights are non-increasing in any walk on H , which means any directed cycle of H must have all edge weights being the same. Moreover, since $\phi(i)$ was defined by breaking ties with lower-indexed clients then all cycles in H have length 2. So, H can be viewed as a collection of connected components each of which is a unicyclic graph consisting of a directed tree with a 2-cycle at the root and all other edges being oriented toward the root. See Figure 4.5 for an example.

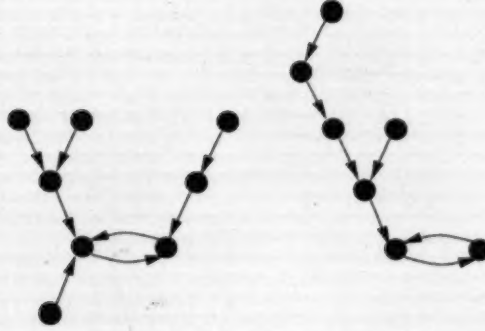
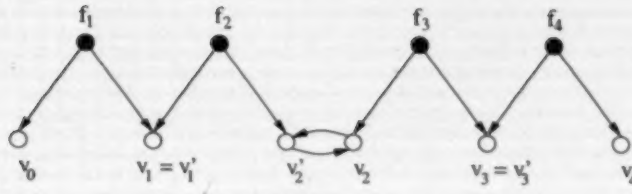


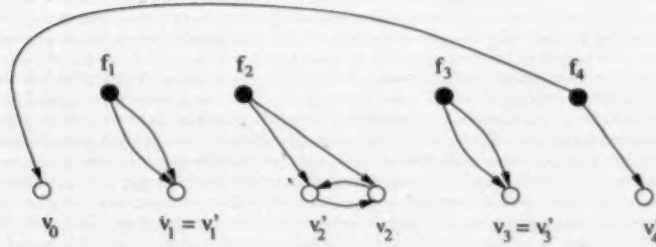
Figure 4.5: An example of an auxiliary graph constructed from the nearest-client function ϕ .

We define two types of *bad* clients. Call each client i a *type 1* client if $x_{ii}^{(3)} = 1$ and for two distinct facilities j and j' , $y_{ji}^{(3)} = y_{j'i}^{(3)} = \frac{1}{2}$. Also, a pair of clients i and i' are called *type 2* if $\phi(i) = i'$, $\phi(i') = i$ and $x_{ii}^{(3)} = x_{i'i'}^{(3)} = \frac{1}{2}$ but two different facilities j and j' are such that $y_{ji}^{(3)} = y_{j'i'}^{(3)} = \frac{1}{2}$. The remaining clients are *good* ones. Recall that the final step of the algorithm will round each facility j to location v with probability $y_{jv}^{(4)}$. Keeping this in mind, we see that type 1 clients are those where $\overline{C}_i^{(3)} = 0$ but i might not receive a facility after the randomized rounding of facilities (and so has to be served at the nearest location with a facility), thus incurring a positive cost increase. We make a modification to the half-integer solution such that this does not happen in the next step. We will bound the expected cost of each client by considering the expected distance a client i has to move along the sequence of locations $\phi^{(0)}(i), \phi^{(1)}(i), \phi^{(2)}(i) \dots$ until it reaches a location that received a facility. Here we define $\phi^{(k)}(i)$ recursively as $\phi^{(0)}(i) = i$ and $\phi^{(k+1)}(i) = \phi(\phi^{(k)}(i))$. The problem with type 2 clients is that they will never reach a location with a facility by iteratively following $\phi^{(k)}(i)$ if both of their half-covering facilities are assigned elsewhere in the randomized rounding phase. If all 2-cycles are guaranteed to receive a facility in the randomized rounding, then all clients will eventually be covered by iteratively following $\phi^{(l)}(i)$ to $\phi^{(l+1)}(i)$ since this sequence eventually reaches a 2-cycle root in H .

We now turn our attention to fixing type 1 and type 2 clients. Begin by setting $(\overline{x}^{(4)}, \overline{y}^{(4)}) \leftarrow (\overline{x}^{(3)}, \overline{y}^{(3)})$. Consider any type 1 client i or type 2 client pairs i and i' with contributing facilities j and j' . We will build a sequence of locations and facilities starting with j . Now, j must be contributing to another location v since it was only contributing $\frac{1}{2}$ to i . If there is a type 1 client i' at v , then continue constructing this sequence with i' followed by the other facility j'' contributing to i' . If there is a type 2 client i' at v , then continue constructing this sequence with i' being followed by $\phi(i')$ and then by the facility j'' contributing to $\phi(i')$. Continue to extend the sequence in this way until a location is reached that is good or until the location of the original type 1 client i is reached. In the former of these two cases, extend the same path from facility j' contributing to the original i . This process forms a sequence consisting only of clients of type 1 or 2 (except, perhaps,



(a)



(b)

Figure 4.6: A sequence of bad clients before (a) and after (b) applying the fixing operation. Each edge represents an assignment of fractional value $\frac{1}{2}$.

the endpoints, if we do not get a cycle) and the facilities which contribute to them. If we do this for every type 1 or 2 client, we get a collection of sequences of which each facility j and location v are adjacent in at most one of them (so that each $y_{jv}^{(4)}$ is represented at most once). We will deal with each of these sequences individually.

Say $v_0, j_1, v'_1, v_1, j_2, v'_2, v_2, \dots, v'_{m-1}, v_{m-1}, j_m, v'_m$ is such a sequence where, for $0 < i < m$, if v_i is a type 1 client we assume $v_i = v'_i$. Finally, if v_0 is of type 1 then we have $v_0 = v'_m$ and if v_0 is of type 2 then $\phi(v_0) = v'_m$. Since we could consider this sequence in the reverse order, without loss of generality, we can assume that:

$$\sum_{i=1}^{m-1} (2d_{j_i, v'_i} + d_{j_i, v_{i-1}}) \leq \sum_{i=1}^{m-1} (d_{j_i, v'_i} + 2d_{j_i, v_{i-1}}) \quad (4.2)$$

Perform the following sequence of updates to fix the type 1 and 2 locations in this sequence (say $v_m = v_0$).

$$\begin{aligned} y_{j_i, v_{i-1}}^{(4)} &\leftarrow 0 & (1 \leq i \leq m) \\ y_{j_i, v'_i}^{(4)} &\leftarrow y_{j_i, v_i}^{(4)} + \frac{1}{2} & (1 \leq i \leq m) \end{aligned}$$

Note that by rule 2 above, if $v'_i = v_i$ (i.e. it is a type 1 client) then we are essentially setting $y_{j_i, v_i}^{(4)} \leftarrow 1$ and if $v'_i \neq v_i$ we will have $y_{j_i, v'_i}^{(4)} = y_{j_i, v_i}^{(4)} = \frac{1}{2}$ (see Figure 4.6 for an example).

There are no more type 1 or type 2 clients remaining left after this update is performed for all of the sequences. It is easy to see that we do not have to change $\mathbb{E}^{(4)}$ values since the amount of facility coverage to a given location does not change. Therefore, we have $\overline{C}^{(4)} = \overline{C}^{(3)}$.

Lemma 4.2.8 $\bar{F}^{(4)} \leq 4\bar{F}^{(3)}$

Proof. For such a sequence $v_0, j_1, v'_1, v_1, j_2, v'_2, v_2, \dots, v'_{m-1}, v_{m-1}, j_m, v'_m$, the cost increase is given by:

$$\begin{aligned}
& \frac{1}{2} \left(\sum_{i=1}^{m-1} (d_{j_i v_i} - d_{j_i v_{i-1}}) + d_{j_m v_0} - d_{j_m v_{m-1}} \right) \\
& \leq \frac{1}{2} \left(\sum_{i=1}^{m-1} (d_{j_i v_i} - d_{j_i v_{i-1}}) + \sum_{i=1}^{m-1} (d_{j_i v_{i-1}} + d_{j_i v_i}) + d_{j_m v_{m-1}} - d_{j_m v_{m-1}} \right) \\
& = \sum_{i=1}^{m-1} d_{j_i v_i} \\
& \leq \sum_{i=1}^{m-1} (d_{j_i v'_i} + d_{v'_i v_i}) \\
& \leq \sum_{i=1}^{m-1} (d_{j_i v'_i} + d_{v'_i v_{i-1}}) \\
& \leq \sum_{i=1}^{m-1} (d_{j_i v'_i} + d_{j_i v'_i} + d_{j_i v_{i-1}}) \\
& \leq \frac{3}{2} \left(\sum_{i=1}^{m-1} d_{j_i v'_i} + d_{j_i v_{i-1}} \right) \\
& = 3 \sum_{i=1}^{m-1} \bar{F}_{j_i}^{(3)}
\end{aligned}$$

The first, second, and fourth inequalities come from the triangle inequality. For each $0 < i < m$, if v'_i is of type 1, then $d_{v'_i v_i} = 0$ and if v'_i and v_i are of type 2, then since $\phi(v'_i) = v_i$ we have that $d_{v'_i v_i} \leq d_{v'_i v_{i-1}}$. In either case, the third inequality holds. The last inequality comes from the assumption in expression (4.2). Summing over all such paths and cycles yields $\bar{F}^{(4)} \leq 4\bar{F}^{(3)}$ because each $y_{j_v}^{(4)}$ variable was updated at most once. $\square$

4.2.5 Randomized Rounding

As mentioned before, our final step is to round each facility j to a location v with probability $y_{j_v}^{(4)}$. Since for each facility j , $\sum_v y_{j_v}^{(4)} = 1$, the expected cost of the facilities after rounding is exactly $\bar{F}^{(4)}$. Naturally, for each client that does not have a facility at its location we send it to the nearest location with a facility after this step is performed. We bound the cost increase due to moving clients in the following lemma. Recall that $\phi^{(k)}(i)$ is defined recursively as $\phi^{(0)}(i) = i$ and $\phi^{(k+1)}(i) = \phi(\phi^{(k)}(i))$ for $k \geq 0$.

Lemma 4.2.9 *The expected cost of the client assignments after the randomized rounding of $\bar{y}^{(4)}$ is at most $4\bar{C}^{(4)}$.*

Proof. For any client i with $x_{ii}^{(4)} = 1$ we have $\bar{C}_i^{(4)} = 0$. Since there are no type 1 clients, then location i is guaranteed to receive a facility so the new cost of this client is still 0. Now assume that client i has $x_{ii}^{(4)} = \frac{1}{2}$. We now prove, for any $k \geq 1$, that:

$$\Pr(\phi^{(k)}(i) \text{ does not receive a facility} \mid \phi^{(l)}(i) \text{ did not receive a facility for all } 0 \leq l < k) \leq \frac{1}{2}.$$

If $x_{\phi^{(k)}(i)\phi^{(k)}(i)}^{(4)} = 1$ then $\phi^{(k)}(i)$ will receive a facility because there are no type 1 clients. Otherwise, let j be the facility with $y_{j\phi^{(k)}(i)}^{(4)} = \frac{1}{2}$. If $y_{j\phi^{(l)}(i)}^{(4)} = 0$ for all $0 \leq l < k$, then the probability that j will be assigned to $\phi^{(k)}(i)$ is exactly $\frac{1}{2}$. If $y_{j\phi^{(l)}(i)}^{(4)} = y_{j\phi^{(k)}(i)}^{(4)} = \frac{1}{2}$ for some $0 \leq l < k$, then it must be that $\phi^{(l)}(i) \neq \phi^{(k)}(i)$. If this were not true, then this implies that $\phi^{(l)}(i)$ is in a cycle of the graph H considered in the previous step. Recall that all cycles of H have length 2 and that there are no type 2 clients, so $\phi^{(l)}(i)$ not receiving a client implies $\phi^{(l+1)}(i)$ must have received one. By the assumption that all $\phi^{(l')}(i)$ do not receive a client for $0 \leq l' < k$, then it must be that $l + 1 = k$ which implies the contradiction $\phi^{(l)}(i) = \phi^{(k)}(i) = \phi^{(l+1)}(i)$ (by construction, the mapping ϕ has no fixed-point). Therefore, $\phi^{(l)}(i) \neq \phi^{(k)}(i)$ and $\phi^{(l)}(i)$ not receiving a facility implies facility j must have been assigned to location $\phi^{(k)}(i)$.

From this, we can see that:

$$\forall k \in \mathbb{N}, \Pr(\phi^{(l)}(i) \text{ do not receive a facility for all } 0 \leq l < k) \leq 2^{-k}.$$

Note that because we do not have a type 2 client pair, for every pair of clients i, i' with $\phi(i) = i'$ and $\phi(i') = i$ (i.e. they form a 2-cycle in H), there must be one facility j with $y_{ji}^{(4)} = y_{ji'}^{(4)} = \frac{1}{2}$. So exactly one of i or i' will be covered by j after the rounding. Any walk on H eventually leads to a 2-cycle (containing the root), of which one location must be assigned a facility, so there is a minimum value k_i such that $\phi^{(k_i)}(i)$ will receive a facility with probability 1 if the previous locations $\phi^{(l)}(i)$ do not receive a facility for $0 \leq l < k_i$.

Also, since the weights of edges of H do not increase in any walk, then it is easy to prove by induction that $d_{i\phi^{(k)}(i)} \leq k \cdot d_{i\phi(i)}$. Thus, the expected cost of serving client i with $x_{ii}^{(4)} = \frac{1}{2}$ is bounded by:

$$\begin{aligned} & \sum_{k=0}^{k_i} D_i \cdot d_{i\phi^{(k)}(i)} \cdot \Pr(\phi^{(k)}(i) \text{ receives a location and } \phi^{(l)}(i) \text{ do not for all } 0 \leq l < k) \\ & \leq \sum_{k=0}^{k_i} k \cdot D_i \cdot d_{i\phi(i)} \cdot \Pr(\phi^{(l)}(i) \text{ do not receive a facility for all } 0 \leq l < k) \\ & \leq D_i \cdot d_{i\phi(i)} \cdot \sum_{k=0}^{\infty} \frac{k}{2^k} \\ & = 2 \cdot D_i \cdot d_{i\phi(i)} = 4\bar{C}_i^{(4)} D_i, \end{aligned}$$

and so the lemma follows. $\square$

4.2.6 Putting it all Together

Working with the modified instance, we have the client/facility costs initially being at most $(\bar{C}, \bar{F})$. After the second step, the new client/facility costs are bounded by $(2\bar{C}, \bar{F} + \bar{C})$. When obtaining the half-integer solution, the costs increase to at most $(2\bar{C} - \tau\Delta, \bar{F} + \bar{C} + \Delta)$ for some constant τ which we will specify shortly. Fixing type 1 and type 2 clients resulted in the cost of the current solution rising to at most $(2\bar{C} - \tau\Delta, 4\bar{F} + 4\bar{C} + 4\Delta)$. Finally, the randomized rounding produced an integer solution to the modified instance with an expected cost of at most $(8\bar{C} - 4\tau\Delta, 4\bar{F} + 4\bar{C} + 4\Delta)$.

However, as detailed in the clustering step, we have to move the demands back to their original locations which is done with a penalty of at most $4\bar{C}$. Thus, the final cost of the algorithm is at most $16\bar{C} + 4\bar{F} + 4(1 - \tau)\Delta$. Choosing the constant τ to be 1 when obtaining the half-integer solution, we see the overall cost of the final integer solution to the original problem being bounded by $16\bar{C} + 4\bar{F} \leq 16 \cdot OPT_f \leq 16 \cdot OPT$.

4.3 Improving to a Deterministic 8-Approximation

In this section, the randomized 16-approximation is improved by considering a better way to relocate assignments to client locations than was done in Section 4.2.2 and improving the analysis of the expected cost increase after the randomized rounding of Section 4.2.5. The resulting algorithm computes, in polynomial time, a feasible solution whose expected cost is no more than 8 times the optimal fractional cost. After presenting these improvements, we discuss how to deterministically compute a feasible solution whose cost is no more than the expected cost of the random solution by applying the method of conditional probabilities in the natural way. To close off this discussion of the deterministic 8-approximation algorithm, we analyze the running time of each step.

4.3.1 Improving Analysis in the Relocation of Coverage

Consider the second step of the algorithm in Section 4.2.2 which consists of relocating the facilities (and the clients with them). Previously, when we moved facility coverage from a location v to the nearest client i' , we also moved all client assignments from v to client i' . Instead of doing that, we will move at most $x_{i'v}^{(2)}$ from each client (assigned to v) and leave any extra client assignment at v . That is, for each client i let $\epsilon_i = \min \{x_{iv}^{(2)}, x_{i'v}^{(2)}\}$ and move this much assignment from v to the location of client i' . For the facilities, define $M = \frac{x_{i'v}^{(2)}}{\sum_j y_{jv}^{(2)}}$ and move this fraction of all facility assignments from v to the location of i' . In other words, for each client i and facility j , perform the following update:

$$\begin{aligned} x_{ii'}^{(2)} &\leftarrow x_{ii'}^{(2)} + \epsilon_i \\ x_{iv}^{(2)} &\leftarrow x_{iv}^{(2)} - \epsilon_i \\ y_{ji'}^{(2)} &\leftarrow y_{ji'}^{(2)} + M \cdot y_{jv}^{(2)} \end{aligned}$$

$$y_{jv}^{(2)} \leftarrow (1 - M) \cdot y_{jv}^{(2)}$$

It may be that after such an update there are still clients assigned to location v . This happens in the case that $x_{i'v}^{(2)} < \max \{x_{iv}^{(2)} \mid i \in C\}$. However, we have moved exactly $x_{i'v}^{(2)}$ total facility assignments from v to the location of i' , so the clients with some assignment to i or v are still covered and client i' now has $x_{i'v}^{(2)} = 0$. We repeat this process on location v with the next closest client i'' that still has $x_{i''v}^{(2)} > 0$ after the re-assignment. Iterate this process until all client assignments to v have been moved to some client location. For each location v and client i' , where the assignments to v were (partially) moved to i' during this process, let $\alpha_{vi'}$ denote the cost of moving the fraction of assignment of i' to v back to i' ; this is $D_{i'}$ times $d_{i'v}$ times the value of $x_{i'v}^{(2)}$ at the time of this movement (note that the client cost for i' decreases by $\alpha_{vi'}$). Let α_v be the sum of all $\alpha_{vi'}$ over all iterations of the above procedure on v . If we consider each client i' that had some portion of the assignment $x_{i'v}^{(2)}$ moved back to $x_{i'i'}^{(2)}$, then the total cost of these clients decreases by α_v and the total facility cost increases by at most α_v . Denote by β_v the cost that all other clients i paid to have their assignments moved from v to some i' location where $i \neq i'$. Let $\alpha = \sum_v \alpha_v$ and $\beta_v = \sum_v \beta_v$ and notice that $\alpha + \beta \leq \bar{C}^{(1)}$ since each fraction of each client is moved at most once. Therefore, the new facility cost $\bar{F}^{(2)}$ is at most $\bar{F}^{(1)} + \alpha$ and the new client cost $\bar{C}^{(2)}$ is at most $\bar{C}^{(1)} + \beta - \alpha$.

4.3.2 Improving the Expected Cost

In the randomized rounding phase, as observed before, for each client i there is a minimum value k_i such that

$$\Pr(\phi^{(k_i)}(i) \text{ receives a facility} \mid \phi^{(l)}(i) \text{ do not receive a facility } 0 \leq l < k_i) = 1.$$

For each $0 \leq l < k_i$, the probability of $\phi^{(l)}(i)$ receiving a facility given that the $\phi^{(l')}(i)$ do not for all $0 \leq l' < l$ is exactly $\frac{1}{2}$. Therefore, the expected cost for client i after the randomized rounding of facilities can be bounded as follows:

$$\begin{aligned} & \sum_{k=0}^{k_i} D_i \cdot d_{i\phi^{(k)}(i)} \cdot \Pr(\phi^{(k)}(i) \text{ receives a location and } \phi^{(l)}(i) \text{ does not for all } 0 \leq l < k) \\ = & \sum_{k=0}^{k_i} D_i \cdot d_{i\phi^{(k)}(i)} \cdot \Pr(\phi^{(l)}(i) \text{ does not receive a facility for all } 0 \leq l < k) \\ & \cdot \Pr(\phi^{(k)}(i) \text{ receives a location} \mid \phi^{(l)}(i) \text{ does not for all } 0 \leq l < k) \\ = & \frac{D_i \cdot d_{i\phi^{(k_i)}(i)}}{2^{k_i}} + D_i \sum_{k=0}^{k_i-1} \frac{d_{i\phi^{(k)}(i)}}{2^{k+1}} \\ \leq & \frac{D_i \cdot k_i \cdot d_{i\phi^{(k_i)}(i)}}{2^{k_i}} + D_i \sum_{k=0}^{k_i-1} \frac{k \cdot d_{i\phi^{(k)}(i)}}{2^{k+1}} \\ = & D_i \cdot d_{i\phi^{(k_i)}(i)} \cdot \left(\frac{k_i}{2^{k_i}} + 1 - \frac{k_i + 1}{2^{k_i}} \right) \end{aligned}$$

```

Cost( $i, l$ )
  if there is a facility assigned to  $v_l$  with fractional value 1 then return  $D_i \cdot d_{iv_l}$ 
  else if there is a facility  $j$  that is assigned to  $v_l$  with fractional value  $\frac{1}{2}$  then
    if  $j$  has a fractional assignment of  $\frac{1}{2}$  to some  $v_j, 1 \leq j < l$  then return  $D_i \cdot d_{iv_l}$ 
    else return  $\frac{1}{2} \cdot D_i \cdot d_{iv_l} + \frac{1}{2} \cdot \text{Cost}(i, l+1)$ 
  else return Cost( $i, l+1$ )

```

Figure 4.7: Recursively computing the expected cost of moving client i to its nearest location with a facility given that all locations $v_j, 1 \leq j < l$ are not assigned a facility.

$$= D_i \cdot d_{i\phi(i)} \cdot \left(1 - \frac{1}{2^{k_i}}\right),$$

where the second last equality can easily be verified by induction. This shows that the expected cost increase for each client i is at most $\bar{C}_i^{(4)}$ (note that the final expression is 0 if $k_i = 0$).

4.3.3 Derandomizing the Algorithm

The method of conditional probabilities can be used to execute the final step of our rounding algorithm deterministically. Since we will move each client i to the nearest location that received a facility after the rounding, we can efficiently compute the expected cost of client i given that some of the facilities are already rounded in the following manner. Begin by ordering the locations $v_1, v_2, \dots, v_{|V|}$ so that $d_{iv_1} \leq d_{iv_2} \leq \dots \leq d_{iv_{|V|}}$. The expected cost of moving client i to its nearest location can be expressed as:

$$\sum_{m=1}^{|V|} D_i \cdot d_{iv_m} \Pr(v_m \text{ receives a facility and } v_l \text{ does not for all } 1 \leq l < m)$$

We can compute the expected cost for client i with the recursive routine of Figure 4.7 as the returned value of initially calling Cost($i, 1$). This routine examines all possible cases of how location v_l can be covered by facilities given that all facility assignments are a half-integer and there are no type 1 or type 2 clients.

Lemma 4.3.1 *The expected cost of client i for a given partial assignment of facilities is Cost($i, 1$).*

Proof. First of all, the recursive routine reaches a base case since all facilities are assigned with weights 1 or $\frac{1}{2}$ to locations in V . Say this base case happens when considering location v_m . We prove, in an inductive fashion, that the value returned when considering location v_l , for $1 \leq l \leq m$, is the expected assignment cost of client i given that all locations $v_{l'}, 1 \leq l' < l$ are not allocated a facility.

Consider the base case of v_m . If v_m has a total facility assignment value of 1, then since there are no type 1 clients it must be that some facility is completely assigned to v_m . If v_m has a weight $\frac{1}{2}$ assignment, then since this is a base case it must be that the facility j partially assigned to v_m was also partially assigned to $v_{l'}$ for some $1 \leq l' < m$. Since j was not assigned to $v_{l'}$ by assumption, it must be assigned to v_m . In either case, the value returned is $D_i \cdot d_{iv_m}$.

Now consider the value computed by calling $\text{Cost}(i, l)$ for $1 \leq l < m$. The first and second cases of the recursive function do not apply because v_l is not a base case. If there is no facility that is partially assigned to v_l then the probability of v_l receiving a facility is 0. Thus, the expected cost is exactly the expected cost of v_{l+1} receiving a facility given that $v_{l'}$ do not for $1 \leq l' \leq l$. If v_l was partially assigned a facility j , then it must be that j is not partially assigned to a previous location $v_{l'}$ for $1 \leq l' < l$ (otherwise v_l is a base case). Thus, the probability of j being assigned to v_l is exactly $\frac{1}{2}$ and the assignment cost for i is $D_i \cdot d_{iv_l}$ in this case. The probability that j is not assigned to v_l is also $\frac{1}{2}$. In this event, the expected assignment cost of i is then recursively computed. $\square$

Now consider each facility j in some order. If j is such that $y_{jv}^{(4)} = 1$ for some v , then we assign j to this location. Otherwise, say v and v' are the two locations such that $y_{jv}^{(4)} = y_{jv'}^{(4)} = \frac{1}{2}$. Try the two cases of assigning j to v and v' and pick the one that produces the least expected cost. When this is done for all facilities, we have an integer assignment of facilities and clients to locations whose cost is at most the expected value of the randomized rounding method.

4.3.4 The Final Analysis

As before, we start with the modified instance which has client/facility costs at most $(\bar{C}, \bar{F})$. The new analysis on the relocating step shows that the cost can be then be bound by $(\bar{C} + \beta - \alpha, \bar{F} + \alpha)$ where $\alpha + \beta \leq \bar{C}^{(1)} \leq \bar{C}$. The cost after obtaining the half integer solution increases to at most $(\bar{C} + \beta - \alpha - \tau\Delta, \bar{F} + \alpha + \Delta)$. Fixing type 1 and type 2 clients has the resulting costs increasing to at most $(\bar{C} + \beta - \alpha - \tau\Delta, 4\bar{F} + 4\alpha + 4\Delta)$. Finally, the new analysis on the expected cost increase, along with the derandomization, shows the integer solution to the modified instance has client/facility costs at most $(2\bar{C} + 2\beta - 2\alpha - 2\tau\Delta, 4\bar{F} + 4\alpha + 4\Delta)$.

As before, move the demands back to their original locations with a cost increase of at most $4\bar{C}$. By choosing $\tau = 2$, we have the final cost of the integer solution being bound by:

$$\begin{aligned} 6\bar{C} + 4\bar{F} + 2\alpha + 2\beta + (4 - 2\tau)\Delta &\leq 8\bar{C} + 4\bar{F} + (4 - 2\tau)\Delta \\ &= 8\bar{C} + 4\bar{F} \\ &\leq 8 \cdot \text{OPT}_f \end{aligned}$$

4.3.5 The Running Time

The running time of the algorithm is primarily dominated by the cost of solving the linear program. Therefore, we begin by analyzing the running time of the remainder of the algorithm and leave the running time of solving the LP as a variable. We only prove that the linear program we solve is of size that is polynomial in the size of the input. From now on, n will denote the number of locations. We will analyze the running time of the algorithm after the pre-processing step of creating a unique location for each facility.

There is an x_{iv} variable for each client i and location v and since we clustered the client pebbles

into a single client (with a demand) on each location, then there are at most n^2 variables x_{iv} . We assumed that each facility resides on each location with no other pebbles so there are also at most n^2 variables y_{jv} . There is a constraint for each client i and each facility j that ensures uniqueness of the destination for the pebble. There is also a constraint for each client/location pair (i, v) that ensures each client is covered. Therefore, the total number of constraints is at most $2n + n^2$. Finally, each constant in the objective function is of the form $D_i \cdot d_{iv}$. D_i is at most $|C|$ and the size of d_{iv} is at most the largest size of any distance in the metric. Finally, each constant appearing as a coefficient or the constant of some constraint is either $-1, 0$ or 1 . Therefore, the size of the linear program we solve is bounded by a polynomial in the size of the input.

The clustering step in Section 4.2.1 relies solely on sorting and a linear-time pass through the clients. Each update can be done in constant time so the running time is $O(n^2)$. In relocating the client coverage, we must consider all of the at most n locations v that do not receive a client. It takes $O(n)$ time to determine if a location has a fractional assignment of clients by simply examining each of the $O(n)$ clients. With the modified algorithm of Section 4.3.1, for each location v we will need to perform at most $|C| \leq n$ relocations of coverage from v to its nearest contributing client. Each of these updates takes $O(n)$ time since we must consider all clients and facilities contributing to v , determine which of these clients is closest and move the appropriate amount of coverage. This results in an $O(n^3)$ algorithm to execute this step. Clearly, the process of removing redundant facility coverage after this step can be done in $O(n^2)$ time.

In getting a half-integer solution in Section 4.2.3, we constructed an auxiliary bipartite graph. As mentioned previously, after each relocation of coverage along an augmenting path, either an edge of the bipartite graph is deleted or a location becomes half-integrally covered. There can be at most $n \cdot |F| \leq n^2$ edges so an upper-bound on the number of iterations is $n^2 + n \in O(n^2)$. Each iteration involves identifying a non-half-integrally covered location which can be done in $O(n^2)$ time. When such a node is identified, a search can be executed from the corresponding vertex in the bipartite graph to find another vertex corresponding to a non-half-integrally covered location. This can be accomplished in $O(n^2)$ time using a standard breadth-first search. Therefore, each of the $O(n^2)$ iterations can be done in $O(n^2)$ time for a total running time of $O(n^4)$. As for obtaining the half-integer solution, the modified instance (by doubling contributions) has $2|F| \leq 2n$ vertices on each side of the bipartition and required $O(n^2)$ time to construct. The minimum weight perfect matching problem in a bipartite graph is solvable by an $O(N^3)$ algorithm where N is the number of vertices on each side of the bipartition (see, e.g., [35]). Therefore, the total running time of getting a half-integer solution is $O(n^4)$.

Since we now have a half-integer solution, we only need to keep track of at most 2 entries for each client or facility. Similarly, for each location, we keep track of the at most 2 facilities that contribute to that location. In each step of modifying the half-integer solution in Section 4.2.4, we found a "bad" client, performed a search to construct a sequence of bad clients and their covering

facilities and then reassigned coverage to fix all of these bad clients. Each bad client is considered in at most one of these iterations and all other clients may be considered in up to two of these iterations (if, say, a good client is the endpoint of two different sequences). For each bad client i , we perform the search in $O(k)$ time where k is the number of clients that will appear in the sequence with i . Performing the update also takes $O(k)$ time. Since each client is considered in at most two of these sequences, adding all k for each iteration results in an amortized $O(n)$ amount of work done in this step.

For the derandomized version of the final step in Section 4.3.3, we consider the $O(n)$ facilities in order. For each facility, we have to consider at most 2 cases of where it will be assigned. When processing each case, we have to determine the expected cost of moving each of the $O(n)$ clients to their nearest facility which can be done if we have a list of the locations sorted by their distance from the client. Given this, the recursive routine takes $O(n)$ time to execute which results in a total of $O(n^2)$ work to determine the expected cost. Computing the sorted list of locations for each client takes a total of $O(n^2 \log n)$ time ($O(n \log n)$ per client). Thus, the derandomized version of the randomized rounding takes $O(n^3)$ time.

The overall running time, apart from solving the linear program, is dominated by the step of fixing non-half-integrally covered locations which takes $O(n^4)$ time. Therefore, the total running time is $O(n^4 + LP)$ where LP is the cost of solving the linear program. A linear program can be solved in polynomial time using, for example, Ye's interior point method [39].

4.4 A Bicriteria Approximation Algorithm

A 5-approximation can be achieved if we allow a slight violation of one of the problem's constraints. Specifically, if we can duplicate each facility pebble at most once (so $\sum_v y_{jv} \leq 2$ for all facilities j) at no extra cost, then we can find a solution with a lower cost.

Consider the algorithm just after the first step of clustering the clients as described in Section 4.2.1. For each client i , let:

$$\alpha_i = \sum_{v: d_{iv} > 2\bar{C}_i^{(1)}} x_{iv}^{(1)}.$$

That is, α_i is the total weight of the assignments of client i to locations that are not in the closed ball R_i . For each client i and location v , define the client i 's assignment to v as:

$$x_{iv}^{(2)} \leftarrow \begin{cases} x_{iv}^{(1)} / (1 - \alpha_i) & \text{if } v \in R_i \\ 0 & \text{otherwise} \end{cases}$$

We still have $\sum_v x_{iv}^{(2)} = 1$ for all clients i , but now all assignments are to locations in the ball R_i . As noted before in Section 4.2.2, these balls are disjoint so no location has two different clients fractionally assigned to it. Since client assignments were moved from locations outside this ball to locations inside this ball, the client costs do not increase, so $\bar{C}^{(2)} \leq \bar{C}^{(1)}$.



Figure 4.8: With D clients at each endpoint, the ratio OPT_2/OPT_1 is D .

Now, the current facility assignments $\bar{y}^{(1)}$ may be infeasible since we increase some of client i 's assignments by $1/(1 - \alpha_i)$ for each i . So, for each location v and facility j , we set j 's assignment to v to be:

$$y_{jv}^{(2)} \leftarrow \begin{cases} y_{jv}^{(1)}/(1 - \alpha_i) & \text{if } v \in R_i \text{ for some } i \\ 0 & \text{otherwise} \end{cases}$$

We have $y^{(2)} \leq 2y^{(1)}$ for each facility j and location v that is covering some client i since $\alpha_i < \frac{1}{2}$ by Lemma 4.2.6, so $\bar{F}^{(2)} \leq 2\bar{F}^{(1)}$. However, it may be that each facility has up to a fraction of 2 being assigned to other locations. Notice that each client i is still covered since the covering constraint of the LP relaxation is $\sum_j y_{jv} \geq x_{iv}$ and both sides were inflated by a factor of $(1 - \alpha_i)^{-1}$ if $v \in R_i$ and, otherwise, $x_{iv}^{(2)}$ was set to 0.

The last step is to recall all client assignment back to their start locations and to move each facility assignment that was covering that client likewise. That is, for each client i , set $x_{ii}^{(3)} = 1$ and $x_{iv}^{(3)} = 0$ for all other $v \neq i$. Then, for each facility j and each location $v \in R_i$ for some client i , move $x_{iv}^{(2)}$ facility coverage from v to i . That is, set $y_{ji}^{(3)} = y_{ji}^{(2)} + x_{iv}^{(2)}$ and $y_{jv}^{(3)} = y_{jv}^{(2)} - x_{iv}^{(2)}$. Since facility coverage follows each client back to its start location and no location has two clients assigned to it in $\bar{x}^{(2)}$, then each client is still covered. The new client cost $\bar{C}^{(3)}$ is 0 since no client is assigned to a location other than its start location. Since each demand $D_i \geq 1$ for any client i then the cost increase for facility j moving from some $v \in R_i$ to the start location of i is $\sum_{i,v} x_{iv}^{(2)} d_{iv} \leq \sum_{i,v} x_{iv}^{(2)} d_{iv} D_i$. Now, in the same spirit as getting the half-integer solution through the matching polytope argument in Section 4.2.3, we have a fractional matching between $|C|$ clients and the now $2|F|$ facilities (if we imagine each original facility as two facilities after the $1/(1 - \alpha_i)$ increase), so we can make this an integer matching of at most the same cost. The final cost of this solution is $\bar{C}^{(2)} + \bar{F}^{(2)} \leq \bar{C} + 2\bar{F}$. However, we must move the demands back as per the instructions in Section 4.2.1 for an added cost increase of $4\bar{C}$. The final cost of this solution is at most $5\bar{C} + 2\bar{F} \leq 5 \cdot OPT_f$.

For a given TM-MFL problem, let OPT_1 denote the optimal integer solution and let OPT_2 denote the optimal integer solution if each facility can be duplicated (replace $\sum_v y_{jv} = 1$ with $\sum_v y_{jv} = 2$ in the IP formulation). We note that the ratio OPT_1/OPT_2 cannot be reasonably bounded which shows that the bicriteria approximation algorithm presented in this section may produce results that are far from optimal if the relaxed facility constraints are used. Consider the graph in Figure 4.8. If the facility is used only one time, then clearly the best solution is to move all clients to the location the facility is starting on for a cost of $2D$. Otherwise, if the facility can be used on two locations, then move one to each of the client locations for a cost of 2. For this problem,

$OPT_1/OPT_2 = D$ which shows that this gap cannot be bound by any constant. If we generalize this example from a path of length 3 to a path of length $2k + 1$ and set $D = 2^{2k+1}$, then we cannot even bound the gap by any sub-exponential function of the number of locations.

Chapter 5

Conclusion

5.1 Summary

The results in this thesis describe the computational complexity of the mobile facility location problem. Both hardness of approximation results and polynomial-time algorithms (either exact or approximate) were presented to exhibit fairly tight complexity-theoretic results on MFL under three of the most popular measures; minimizing the maximum distance a pebble moves, the total distance travelled by all pebbles, and the number of pebbles that must be moved.

It turned out that minimizing the number of pebbles to be moved can be solved exactly by a fairly simple polynomial-time algorithm. This isn't too surprising since Demaine *et. al.* (in [13]) discovered an exact algorithms and a PTAS for minimizing the number of pebbles moved in two of the movement problems they considered. However, the minimum cardinality measure was the only one that we considered for MFL that had an efficient and exact algorithm.

Minimizing the maximum any pebble moved also had a simple algorithm (as noted in [13]) that constructs a 2-approximate solution. We addressed the question of improving this approximation guarantee by proving that there is no α -approximation algorithm for MM-MFL unless $P = NP$ for any $1 \leq \alpha < 2$. Essentially, this result completely classifies the approximability of MM-MFL if $P \neq NP$.

Before digging into the main result for TM-MFL, we presented a pseudo-polynomial-time algorithm for solving the total movement version in trees. This method was presented as a dynamic programming algorithm as the tree was traversed in a post-order fashion. From this, we used a previous result on approximating arbitrary metrics by tree metrics to develop a randomized $O(\log n)$ -approximation algorithm that runs in polynomial time in the number of locations and clients.

Finally, in what we consider the most interesting result of the thesis, we presented an approximation algorithm for minimizing the total movement of all pebbles that produces a feasible solution whose cost is within a constant factor of the optimal solution. This was first motivated by proving that TM-MFL cannot be approximated beyond a certain constant unless $P = NP$ and that the general k -median problem with demands can be solved approximately by obtaining an approximate

solution to a specific instance of TM-MFL that is related to the k -median instance. Since there is no 1.735-approximation for the k -median problem unless $NP \subseteq DTIME(n^{O(\log \log n)})$ [24], then this same hardness result holds for TM-MFL.

The actual approximation algorithm for TM-MFL was presented in stages. Through a five step process, we showed how any fractional solution to a given instance with cost T can be turned into an integer solution with expected cost at most $16 \cdot T$ through the use of randomized rounding. Upon a minor improvement to one step and a tighter analysis of another step, we showed that we can produce an integer solution with expected cost $8 \cdot T$. It turned out that the randomized rounding step can be derandomized to obtain a deterministic algorithm with the same approximation guarantee. We also noted a 5-approximation algorithm that produces an infeasible solution in the sense that each facility pebble can be duplicated at most once.

5.2 Directions for Future Work

The most natural open problem related to the results in this thesis is improving the approximation guarantee of the algorithm for TM-MFL. That is, we have an 8-approximation for the problem but a hardness of only $1 + 2/e - \epsilon$ for all $\epsilon > 0$ under the plausible assumption $NP \not\subseteq DTIME(n^{O(\log \log n)})$. Our approximation algorithm has a ratio that is roughly 4.6 times larger than the hardness bound, so there is room for improvement to either the upper or lower bound. Also, our approximation method, while polynomial in running time, relies on finding an optimal solution to a linear program. An approximation algorithm that avoids the overhead of solving an LP is desirable.

It seems doubtful that we can obtain an approximation ratio better than 4 or 5 by following the framework of our algorithm because of the clustering step in Section 4.2.1. Therefore, we will likely have to look at other methods altogether for a significant improvement to the approximation ratio. Currently, the best approximation algorithm for k -median uses the technique of local search [3]. This technique was also successfully applied to the uncapacitated facility location problem and some of its variants such as the capacitated facility location problem [3] and facility location with hierarchical opening costs [37]. Perhaps we could see some success applying local search to TM-MFL as well. Other successful methods for approximating the uncapacitated facility location problem deal with the linear programming relaxation of the problem and its dual and do not require explicitly solving an LP. The LP-relaxation of TM-MFL produces a dual that seems more difficult to work with than the dual of the uncapacitated facility location LP-relaxation, but this does not necessarily mean that popular methods (such as primal-dual or dual fitting) are doomed to failure. Specifically, the dual of

the LP-relaxation to TM-MFL as described in Section 4.1, before converting to standard form, is:

$$\begin{aligned}
 &\text{maximize :} && \sum_{i \in C} \alpha_i + \sum_{j \in F} \gamma_j \\
 &\text{such that :} && \alpha_i - \beta_{iv} \leq D_i d_{iv} \quad \forall i \in C, v \in V \\
 &&& \gamma_j + \sum_{i \in C} \beta_{iv} \leq d_{jv} \quad \forall j \in F, v \in V \\
 &&& \beta_{iv} \geq 0 \quad \forall i \in C, v \in V
 \end{aligned}$$

The demands D_i of a client i can be thought of as a weight for moving client i . One possible generalization of TM-MFL is to assign a weight w_j to each facility j as well so that the cost of moving j to v is $w_j \cdot d_{jv}$. While most steps of our algorithm can be easily modified to work in this more general case, adapting the step of relocating coverage to clients in Section 4.2.2 seems to be difficult. Another possible variant is to assign a capacity C_j to each facility j meaning that only C_j clients can be served by facility j which is similar to the capacitated facility location problem just as TM-MFL is similar to UFL.

As mentioned in [13], many classical optimization problems can be defined in this movement setting. To the best of our knowledge, the only problems that have been considered in the movement setting are the mobile facility location problems discussed in this thesis and the problems presented in [13]. One interesting and unexplored problem is the variant when pebbles reside on edges of a graph and they must be moved around to other edges so that the deletion of all edges with a pebble will increase the number of connected components of the graph. Another possible problem is moving pebbles around vertices of a graph so that the subgraph induced by the vertices with pebbles contains k disjoint paths between two specified nodes s and t . The case where $k = 1$ has already been considered in [13].

Bibliography

- [1] N. Alon, R. Karp, D. Peleg, and D. West, *A graph-theoretic game and its application to the k -server problem*, SIAM J. Comput, 24 (1995), pp. 78–100.
- [2] N. Alon, D. Moshkovitz, and M. Safra, *Algorithmic construction of sets for k -restrictions*, ACM Transactions on Algorithms (TALG) 2 (2) (2006), pp. 153–177.
- [3] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit, *Local search heuristics for k -median and facility location problems*, SIAM J. Comput, 33 (2004), pp. 544–562.
- [4] Y. Bartal, *Probabilistic approximations of metric spaces and its algorithmic applications*, in Proc. of the 37th annual IEEE Symp. on Foundations of Comp. Sci. (FOCS), 1996, pp. 184–193.
- [5] Y. Bartal, *On approximating arbitrary metrics by tree metrics*, in Proc. of the 29th annual ACM Symp. on Theory of Computing (STOC), 1997, pp. 161–168.
- [6] J.L. Bredin, E. Demaine, M.T. Hajiaghayi, and D. Rus, *Deploying sensor networks with guaranteed capacity and fault tolerance*, in Proc. of the annual 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC), 2005, pp. 309–319.
- [7] M. Charikar, C. Chekuri, A. Goel, and S. Guha, *Rounding via trees: deterministic approximation algorithms for group Steiner trees and k -median* in Proc. of the 30th annual ACM Symp. on Theory of Computing (STOC), 1998, pp. 114–123.
- [8] M. Charikar and S. Guha, *Improved combinatorial algorithms for facility location and k -median problems*, in Proc. of the 40th Annual IEEE Symp. on Foundations of Comp. Sci. (FOCS), 1999, pp. 378–388.
- [9] M. Charikar, S. Guha, D. Shmoys, and E. Tardos, *A constant factor approximation algorithm for the k -median problem*, in Proc. of the 31st annual ACM Symp. on Theory of Computing (STOC), 1999, pp. 1–10.
- [10] F. Chudak and D.B. Shmoys, *Improved approximation algorithms for uncapacitated facility location problem*, SIAM J. Comput, 33 (2003), pp. 1–25.
- [11] V. Chvátal, *A greedy heuristic for the set-covering problem*, Mathematics of Operations Research 4 (3) (1979), pp. 233–235.
- [12] V. Chvátal, *Linear Programming*, Freeman, New York, 1983.
- [13] E. Demaine, M. Hajiaghayi, H. Mahini, A. Sayid-Roshkar, S. Oveisgharan, and M. Zadimoghaddam, *Minimizing movement*, In Proc. of the 18th annual ACM-SIAM Symp. on Discrete Algorithms (SODA), 2007, pp. 258–267.
- [14] J. Fakcharoenphol, S. Rao, and K. Talwar, *A tight bound on approximating arbitrary metrics by tree metrics*, JCSS, 69 (2004), pp. 485–497. Preliminary version in Proc. 35th Annual ACM Symp. on Theory of Computing (STOC), 2003.

- [15] U. Feige, *A threshold of $\ln n$ for approximating set-cover*, in Proc. of the 28th annual ACM Symp. on Theory of Computing (STOC), 1996, pp. 314–318.
- [16] Z. Friggstad and M. Salavatipour, *Minimizing Movement in Mobile Facility Location Problems*, submitted to the 19th annual ACM-SIAM Symp. on Discrete Algorithms (SODA), 2008.
- [17] S. Guha and S. Khuller, *Greedy strikes back: improved facility location algorithms*, in Proc. of the 9th annual ACM-SIAM Symp. on Discrete Algorithms (SODA), 1998, pp. 649–657.
- [18] J. Hastad, *Some optimal inapproximability results*, in Proc. of the 29th annual ACM Symp. on Theory of Computing (STOC), 1997, pp. 1–10.
- [19] D. Hochbaum, *Heuristics for the fixed cost median problem*, Mathematical Programming, 22 (1982), pp. 148–162.
- [20] D. Hochbaum and D. Shmoys, *A unified approach to approximation algorithms for bottleneck problems*, J. ACM, 33 (1986), pp. 533–550.
- [21] T-R. Hsiang, E.M. Arkin, M. Bender, S. Fekete, and J.S. Mitchell, *Algorithms for rapidly dispersing robot swarms in unknown environments*, in Proc. of the 5th Workshop on Algorithmic Foundations of Robotics (WAFR), 2002, pp. 77–94.
- [22] W. Hsu and G. Nemhauser, *Easy and hard bottleneck location problems*, Disc. Applied Math., 1 (1979), pp. 209–216.
- [23] K. Jain, M. Mahdian, E. Markakis, A. Saberi, and V.V. Vazirani, *Approximation algorithms for facility location via dual fitting with factor-revealing LP*, J. ACM, 50 (2003), pp. 795–824.
- [24] K. Jain, M. Mahdian, and A. Saberi, *A new greedy approach for facility location problems*, in Proc. of the 34th annual ACM Symp. on Theory of Computing (STOC), 2002, pp. 731–740.
- [25] K. Jain and V.V. Vazirani, *Approximation algorithms for metric facility location and kmedian problems using the primal-dual schema and Lagrangian relaxation*, J. ACM, 48 (2001), pp. 274–296.
- [26] R. Karp, *Reducibility among combinatorial problems*, In Raymond E. Miller and James W. Thatcher, editors *Complexity of Computer Computations*, 85–103, Plenum Press, 1972.
- [27] M. Korupolu, C. Plaxton, and R. Rajamaram, *Analysis of a local search heuristic for facility location problems*, in Proc. of the 9th annual ACM-SIAM Symp. on Discrete Algorithms (SODA), 1998, pp. 1–10.
- [28] A. Kuehn and M. Hamburger, *A heuristic program for locating warehouses*, Management Science, 9 (1963), pp. 643–666.
- [29] J-H. Lin and J. Vitter, *Approximation algorithms for geometric median problems*, Inform. Proc. Lett., 44 (1992), pp. 245–249.
- [30] J-H. Lin and J. Vitter, *ϵ -approximations with minimum packing constraint violation*, in Proc. of the 24th annual ACM Symp. on Discrete Algorithms (STOC), 1992, pp. 771–782.
- [31] M. Mahdian and M. Pál, *Universal facility location*, in Proc. of the 11th annual European Symp. on Algorithms (ESA), 2003, pp. 409–421.
- [32] M. Mahdian, M. Markakis, A. Saberi, and V. Vazirani, *A greedy facility location algorithm analyzed using dual fitting*, in Proc. 5th International Workshop on Randomization and Approximation Techniques in Computer Science, vol. 2129 in Lecture Notes in Computer Science, pp. 127–137, Springer-Verlag, New York, 2001.
- [33] M. Mahdian, Y. Ye, and J. Zhang, *Approximation algorithms for metric facility location problems*, SIAM J. Comput., 36 (2006), pp. 411–432.

- [34] M. Mahdian, Y. Ye, and J. Zhang, *Improved approximation algorithms for metric facility location problems*, in Proc. of 5th annual International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX), 2002, pp. 229–242.
- [35] A. Schrijver, *Combinatorial Optimization - Polyhedra and Efficiency*, Springer-Verlag, New York, 2005.
- [36] D. Shmoys, É. Tardos, and K. Aardal, *Approximation algorithms for facility location problems*, in Proc. of the 29th annual ACM Symp. on Theory of Computing (STOC), 1997, pp. 265–274.
- [37] Z. Svitkina and É. Tardos, *Facility location with hierarchical facility costs*, in Proc. of the 17th annual ACM-SIAM Symp. on Discrete Algorithms (SODA), 2006, pp. 153–161.
- [38] M. Sviridenko, *An improved approximation algorithm for the metric uncapacitated facility location problem*, in Proc. of 9th Inter. Conf. on Integer Programming and Combinatorial Optimization (IPCO) 2002, pp. 240–257.
- [39] Y. Ye, *An $O(n^3 L)$ potential reduction algorithm for linear programming*, Mathematical Programming, 50 (1991), pp. 239–258.